Calhoun
Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1975

A sequential strain monitor and recorder for use in aircraft fatigue life prediction.

Vidrine, David Matthew

http://hdl.handle.net/10945/21031

Downloaded from NPS Archive: Calhoun

DUDLEY
KNOX
LIBRARY

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

A SEQUENTIAL STRAIN MONITOR AND RECORDER FOR USE IN AIRCRAFT FATIGUE LIFE PREDICTION

David Matthew Vidrine

Library
Naval Postgraduate School
Monterey, California

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A Sequential Strain Monitor and Recorder for Use in Aircraft Fatigue Life Prediction

by

David Matthew Vidrine

June 1975

Thesis Advisor: G. H. Lindsey

Approved for public release; distribution unlimited.

T167521

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
A Sequential Strain Monitor and Recorder for Use in Aircraft Fatigue Life Prediction

5. TYPE OF REPORT & PERIOD COVERED
Engineer's Thesis; June 1975

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
David M. Vidrine

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

12. REPORT DATE
June 1975

13. NUMBER OF PAGES
109

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)
Naval Postgraduate School
Monterey, California 93940

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Aircraft fatigue life prediction.

Sequential strain monitor.

Microprocessor applications.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A new system for monitoring, analyzing and recording aircraft structural loads is put forth. The sensors are resistance strain gages which feed a differential analog signal to a multiplexer/analog-digital converter. The digitized strain levels are read by a microcomputer, analyzed and recorded on a digital, cassette recorder, when fatigue-significant events are detected. Automated data retrieval and preparation with this proposed system should be straightforward and serve to alleviate much of the manual handling now required. Suggestions for future research include the fabrication and

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
(Page 1) S/N 0102-014-6601 |

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

testing of a more advanced prototype of the sequential strain monitor system.

A Sequential Strain Monitor and Recorder for Use in Aircraft Fatigue Life Prediction

by

David Matthew Vidrine
Lieutenant Commander, United States Navy
B.S., Naval Postgraduate School, 1973
M.S.A.E., Naval Postgraduate School, 1974

Submitted in partial fulfillment of the
requirements for the degree of

AERONAUTICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
June 1975

Library
Naval Postgraduate School
Monterey

ABSTRACT

A new system for monitoring, analyzing and recording aircraft structural loads is put forth. The sensors are resistance strain gages which feed a differential analog signal to a multiplexer/analog-digital converter. The digitized strain levels are read by a microcomputer, analyzed and recorded on a digital, cassette recorder, when fatigue-significant events are detected. Automated data retrieval and preparation with this proposed system should be straightforward and serve to alleviate much of the manual handling now required. Suggestions for future research include the fabrication and testing of a more advanced prototype of the sequential strain monitor system.

TABLE OF CONTENTS

I. INTRODUCTION.. 9
II. CURRENT FATIGUE LIFE PREDICTION PROGRAM........ 11
    A. ASSUMPTIONS INHERENT IN CURRENT DATA....... 11
    B. DATA EXTRACTION AND HANDLING.............. 12
III. ATTEMPTS TO FIND BETTER METHODS.............. 14
    A. SCRATCH GAGE.............................. 14
    B. S/N FATIGUE LIFE GAGE..................... 15
    C. STRAIN LEVEL COUNTERS..................... 15
IV. PROPOSED SOLUTION............................. 17
V. SYSTEM DESCRIPTION............................ 17
    A. A/D CONVERTER............................. 17
    B. PL-805 MICROCOMPUTER...................... 18
    C. READ-WRITE TAPE RECORDER.................. 19
    D. KEYBOARD.................................. 20
    E. HEXADECIMAL DISPLAYS...................... 20
    F. DIGITAL INPUT SELECT CARD................. 20
    G. SYSTEM SOFTWARE........................... 21
        1. FLD................................... 21
        2. FLTR.................................. 22
        3. Service Modules....................... 22
            a. JUMP.............................. 22
            b. LGAP.............................. 23
            c. SGAP.............................. 23
            d. CLK............................... 23
            e. KEY............................... 23
            f. RCDR.............................. 24
    H. DATA EXTRACTION........................... 24
    I. INTERFACE WIRING TABLES................... 25
VI. SUGGESTIONS FOR FUTURE RESEARCH.............. 33
VII. CONCLUSIONS.................................. 34

LIST OF TABLES

I. KEY TO SYMBOLS USED IN INTERCONNECTION TABLES.. 26
II. INPUT PORT 0-KEYBOARD........................ 26
III. INPUT PORT 0-DISC........................... 27
IV. INPUT PORT 1-MEMODYNE/DAS-16................. 27
V. INPUT PORT 2-DAS-16........................... 28
VI. INPUT PORT 3-DAS-16.......................... 28
VII. OUTPUT PORT 0-DISPLAY 0..................... 29
VIII. OUTPUT PORT 1-DISPLAY 1.................... 29
IX. OUTPUT PORT 2-DAS-16/DISC 30
X. OUTPUT PORT 3-DAS-16/MEMODYNE................. 30
XI. CLOCK-DISC................................... 31
XII. POWER DISTRIBUTION.......................... 32

LIST OF DRAWINGS

1. System Block Diagram.............................. 35
2. Continuous Signal vs "Filtered" Data............ 36
3. "Filtered" Data Alone (Same input)............. 37
4. FLTR Logic Flow Diagram.......................... 38
5. Tabular Data Output.............................. 39

ACKNOWLEDGEMENTS

This author wishes to express his profound appreciation to Lt. James W. Sturges for his indespensable assistance in this project. It was through his suggestion that the MIDAS and sequential strain monitor systems were fused and his intimate knowledge of solid state electronics and hardware was the keystone which permitted the concept to take shape. Lt. Sturges also rendered much assistance in the preparation of this document in the form of details of system information and description.

A debt of gratitude is also owed to Mr. T. B. Dunton, Supervisory Aerospace Technician, who, from an expertise born of long experience, rendered invaluable assistance in the planning and fabrication of this system. He was also instrumental in expediting the acquisition of several vital items of hardware without which the project could not have been completed in the time available.

The patient and rapid response of the model makers of the metal and wood shops and the assistance of the technicians of the electronics shops is also acknowledged with deep appreciation. Their rapid, cheerful response to short-lead-time requests was indespensable to the timely completion of this project.

To Professor Lindsey, my thesis advisor, go my thanks for initially suggesting the general area of fatigue, thus planting the seed which led to this concept.

A very special thank-you must go to my long-suffering wife, Barrie, without whose loving patience and encouragement this endeavor would have been doomed to failure. She endured the many nights and weekends of husbandless existence without complaint and carried more than her share of the domestic burdens while this work was being accomplished.

I. INTRODUCTION

Aerospace structures constitute a compromise between long fatigue life and minimum weight. The driving need for minimum weight to achieve performance specifications forces the engineer to design to the very limit of the material capability; thus producing a fatigue-limited structure. Because usage of each aircraft varies widely, the Navy has turned to monitoring each aircraft for fatigue life expenditure. In addition, the extremely high cost of today's weapons systems makes it imperative that they be used over as long a service life as possible. These extended service lives also produce problems of fatigue which must be handled by accurate monitoring and periodic update, repair and/or replacement of structures. Herein is proposed a new approach to the monitoring of aircraft fatigue life which, predicated upon sequential strain history, is believed will serve to perform that task much better.

Apart from the very obvious safety aspects of the fatigue problem, there are several additional considerations which motivate the development of a more accurate fatigue life prediction technology. Economic considerations are increasingly important in the ever more restricted fiscal environment, especially with the phenomenal growth in the cost of procuring a new weapons system. The high cost of replacing an aircraft which has experienced a catastrophic inflight structural failure is only one aspect of the overall cost of maintaining a weapon in the inventory. Present practice is to replace or rework fatigue-critical portions of a structure at various milestones in its service life, or when some predefined level of structural fatigue life expended is reached. Premature retirement or rework of an aircraft

based upon the conservative estimates required by the present method of monitoring flight loads can be extremely costly, both in terms of dollars and lost effectiveness due to aircraft down time.

A specific case in point is the periodic replacement of the A-6 wing at some fixed point in the planned life, with each such replacement costing approximately one-half million dollars. The cost of a sequential strain monitor system has been very roughly estimated at $3000.00; thus, the prevention of even one such needless change would compensate for the cost of many such systems. It would not require a large number of such savings to justify the costs involved in the procurement of a sequential strain monitor system and the associated support equipment.

II. CURRENT FATIGUE LIFE PREDICTION PROGRAM

The current Fatigue Life Prediction (FLP) program conducted by the Naval Air Development Center was begun in 1969 and represented a significant step forward in the responsible handling of Navy aircraft. Recent technological advances have made it possible to gather much more accurate and meaningful flight data and a modernization of the FLP is currently being studied. The program as it exists today is described briefly below.

A. ASSUMPTIONS INHERENT IN CURRENT DATA

The current sensor used to monitor flight loads on Naval aircraft is an accelerometer placed near the aircraft center of gravity, and has been in general use since 1962. The placement of this sensor can be only approximate, as the center of gravity varies widely with the fuel and ordnance load carried and changes markedly during the course of a single flight, as fuel is burned and ordnance expended. This fluctuation in the location of the center of gravity changes the relationships assumed to exist between the sensed acceleration and the load experienced by the structure. Uncertainty concerning the real nature of the acceleration data exists, for the type of maneuver producing the acceleration is not known and therefore must be assumed to be the worst possible from a fatigue damage point of view.

Since normal acceleration itself cannot indicate the actual load experienced by the structure, an assumption must be made as to the gross weight to be used in converting acceleration to structural load. It is known that the assumed average will be in error for the general case so it is necessary to make a conservatively high estimate,

usually resulting in more fatigue damage being attributed to the structure than is actually experienced.

The damage calculations for all fleet aircraft, with the exception of the Grumman A-6, employ a modified Palmgren-Miner damage sum without accounting for sequence. The A-6 calculations do use a sequence-dependent damage rule; however, another conservatism creeps into these calculations. Since nothing is known of loading sequence in the present system, the worst possible sequence must be assumed. The penalty for assuming the worst sequence is a reduction in life of a factor that may be as high as one fourth or one fifth [Ref 8]. The judicious elimination of this conservatism alone could conceivably keep an aircraft flying safely long after present criteria demand its retirement. The present conservatism in assumptions is made necessary by the large uncertainties in the data acquired. The reduction of data uncertainties would allow the level of conservatism to be safely reduced so that more efficient utilization of costly aircraft can be achieved.

B. DATA EXTRACTION AND HANDLING

The current system of data extraction and handling requires manual interaction and involvement at every step of the process. It begins with the monthly recording of the acceleration level counter readings by squadron maintenance personnel, followed by the transcription of these data to postcards and the mailing of these cards to NADC. At NADC the cards must be individually examined for format and content; then, when the cards have been verified, the data contained are hand keyed to punched cards, which must themselves be hand checked for accuracy. The next step is the transcription of the cards to magnetic tape and the printout of the data to make yet another manual check for completeness and accuracy. When all known errors have been

corrected, the magnetic tape is ready to be used in the fatigue life prediction computer program.

With the approximately 3000 aircraft already monitored in this fashion, three people are required on a full time basis and a fourth is partially occupied. With plans to extend the program to cover more aircraft, a larger number of personnel may be required in the future. In addition to the high workloads involved in the preparation and verification of data from hand written records, there is the problem of lost data. It is not unusual that a postcard becomes lost in the mail and the data are unavailable for that aircraft for that month. This is not an irrevocable loss, if the aircraft instrument is not changed in the interim, because the counters are cumulative. If the recorder has been changed in the intervening period, an average of the valid data obtained for similar aircraft is used in place of the missing data.

III. ATTEMPTS TO FIND BETTER METHODS

Many attempts have been made to produce an economical, accurate and trouble-free fatigue life sensor. Some have enjoyed more success than others in producing usable information, but none has gained wide acceptance or seen wide use. The Air Force has sponsored experiments with scratch gages [Ref 4] and the Navy has investigated the effectiveness of S-N fatigue life gages in specimen fatigue tests [Ref 7]. A brief discussion of these approaches and the current Navy strain level counter system is given below.

A. SCRATCH GAGE

The scratch gage has the advantages of simplicity, very low procurement and maintenance cost, sequential information and permanent records but there are several drawbacks associated with this system. The size of the instrument, with a gage length of approximately six inches [Ref 4], engenders considerable difficulty in installation and placement. Given that the gage can be installed in a location of interest, there is the difficulty associated with data extraction, since optical readers of high precision and magnification are needed to interpret the scratch records. Using current technology, this characteristic would necessitate extensive manual handling and preparation of data with the attendant difficulties already discussed. The perfection of an automatic optical reader and transcriber would do much to make the scratch gage a viable, inexpensive fatigue life system.

B. S/N FATIGUE LIFE GAGE

The concept of the permanently installed, periodically checked, fatigue life gage is appealing in its simplicity, and much effort has been expended toward producing a practical S/N gage; however, results achieved to date have been less than satisfactory. Tests were conducted on a device of this type at NADC in order to determine if such a device would answer the needs of the Navy fatigue life program. It was the conclusion of the evaluators that the S/N gage was capable of showing a qualitative relationship between resistance change and fatigue damage accumulated, but that no quantitative relationship could be established reliably. The conclusion was that the S/N fatigue life gage, at least at present, is unsuitable and the accuracy of data produced is unacceptable [Ref 7].

C. STRAIN LEVEL COUNTERS

The strain level counter device currently being studied by NADC has several characteristics to recommend its use. The data received by the instrument are actual strain information instead of approximate load derived from normal acceleration and an assumed gross weight. It is comparable in cost to present systems and uses existing counter boxes to record data. The disadvantages of this approach, although it represents a notable step forward, are that it is limited to only one channel of information coming from a single transducer, provides no sequence information and still requires manual data extraction and preparation with the attendant problems discussed earlier.

IV. PROPOSED SOLUTION

From the author's investigations into the current techniques of calculating fatigue damage, it seems readily apparent that a need exists for a light, compact, sequential strain recorder to replace the current systems used for fatigue monitoring. Such a system could provide the data needed to more efficiently utilize today's multi-million dollar weapons systems. In order to extend the FLP program to cover virtually all navy aircraft, the data extraction, preparation and handling should be as automated as possible.

What is proposed then is a multichannel system to monitor and analyze structural loads, and then record strain data in a compressed form containing only fatigue-significant events. A prototype system, which fills the functional needs discussed above, has been developed and is currently being tested by the Aeronautics Department of the Naval Postgraduate School. It was developed in conjunction with, and as part of, a general data gathering system called MIDAS (Microprogrammable Integrated Data Acquisition System). During the design and fabrication of the MIDAS/sequential strain monitor system, it was found that the requirement to "filter" the incoming data, decide what is pertinent and control the recording requires a computer of some description, as the logic involved makes a hardware approach extremely difficult. The prototype system is described below.

V. SYSTEM DESCRIPTION

The prototype sequential strain monitor consists of several hardware modules purchased as units and interfaced to produce the integrated system, which occupies approximately two cubic feet and weighs thirteen pounds. A dedicated, follow-on prototype could be considerably reduced in both bulk and weight, a matter discussed in the Suggestions for Future Research section of this document. A block diagram of the MIDAS/sequential strain monitor system is shown in figure 1.

Incoming analog signals are fed to a 16 - channel multiplexer/analog-digital converter, which converts the analog voltage level to an eight bit binary representation. The converted voltage level is read by a microcomputer, which performs the data analysis and issues control commands to the peripheral equipment. The data are stored temporarily in an output buffer in the random access memory (RAM) of the microcomputer, but then fed to the cassette tape recorder when the buffer is full. The tape recorder is an incremental, digital cassette recorder, which uses the industry standard "PHILLIPS" cassettes containing 300 feet of tape and having a capacity to store 180,000 fatigue events.

There is a keyboard for program and/or parameter entry and hexadecimal displays to aid in user prompting and for program auditing. A detailed description of the various hardware modules is given below.

A. A/D CONVERTER

The A/D converter is a Datel Systems Inc. DAS-16-L12B, which can monitor up to 16 channels of analog information in the range minus-five-to-plus-five volts. Two modes of

operation, random and sequential, allow channels to be monitored in sequence or separately addressed, as desired. In the random mode the channel number must be supplied to the DAS-16 along with a strobe pulse to select the channel to be converted. In the sequential mode, channels are monitored in sequence, beginning with channel 1 when a reset command is received. The sequential mode can be "short-counted" by a hardware change, or the short-count can be accomplished by sending a reset command at the appropriate time.

Maximum convert time for a full scale slew of the input signal is advertised as 40 microseconds. An End-of-Convert signal is output when conversion is complete and the data is latched on the data output lines. A detailed description of the interface connections is given in the wiring tables.

B. PL-805 MICROCOMPUTER

The Pro-Log Corporation PL-805 is a complete microcomputer system contained on five printed circuit cards. It utilizes the Intel Corporation 8008-1 microprocessor, which has an instruction set consisting of 230 separate commands. The instructions are broken down into seven functional categories.

Register - (Load and count)

Memory - (Load and store)

Arithmetic - (Add and subtract)

Logical - (AND, OR, XOR, Compare, and rotate)

Program Address Control - (Decisions)

Interrupt Control

Input/Output

A complete list of the microprocessor instructions is included in the appendix.

The PL-805 has four input and four output ports of eight lines each for a total of 32 input and 32 output lines. The ports are used for exchanging data and status/control information among system components. The specific port and line assignments are detailed in the wiring tables.

The PL-805 has a capacity for 2048 words of Programmable Read Only Memory (PROM) and 4096 words of Random Access Memory (RAM) organized into "pages" of 256 words each. PROM is used for permanently resident programs and is non-volatile whereas RAM is volatile, allowing both read and write operations for temporary program storage or data buffering. The appendix contains detailed hardware descriptions and schematic drawings. The details of interface connections are contained in the wiring tables.

C. READ-WRITE TAPE RECORDER

The tape recorder is a Memodyne Corporation Model BR-103 consisting of a Model 98 tape transport with rewind capacity, End-of-Tape/Beginning-of-Tape sensing, Motor Drive Card, Write Step Card and a Read Amplifier Card. A Read Oscillator card was manufactured locally to supply a 360 Hz. signal to the Motor Drive Card for read operations. The four - phase stepping motor provides precise control of tape motion at 120 steps per inch, permitting asynchronous storage and retrieval of data without producing large gaps of blank tape.

Two waveforms are required to write data to tape: the Motor Clock to step the drive motor and the Write Clock to trigger the writing of a flux change on the tape. Information is stored on the tape in two channels such that a flux change on one channel indicates a logical "1" and a flux change on the other indicates logical "0." Both the Write Clock and the Motor Clock are generated in software for write operations, permitting extremely precise control of the recorder.

For read operations, the Motor Clock is generated by the Read Oscillator card while the microcomputer monitors the recorder output for data. The separate oscillator for the read operation is necessary to avoid possible dropout of information in the read mode while generating a Motor Clock in software. Data must be supplied to and extracted from the recorder serially, a feature which allows great flexibility in data formating. Details of the interface connections are given in the wiring tables.

D. KEYBOARD

The keyboard contains the numerals 0 through 9 and the characters A through F to represent the hexadecimal number system. The "Word Entry" key sends a signal to the microcomputer prompting data entry.

E. HEXADECIMAL DISPLAYS

Four Hewlett Packard hexadecimal displays are mounted immediately above the keyboard for use in prompting the user and for program review/verification. Binary output from the microcomputer is internally converted to hexadecimal characters and displayed on a matrix of light emitting diodes.

F. DIGITAL INPUT SELECT CARD

The DISC is a locally constructed device to multiplex several digital signals to Input Port 0 of the microcomputer. The present installation multiplexes the keyboard and digital clock outputs to Port 0, but such a device would not be needed in a dedicated sequential strain monitor system. The interested reader is referred to Reference 9 for a more detailed description.

G. SYSTEM SOFTWARE

The system software consists of the Fatigue Life Data (FLD) program and several supporting service modules which control the interface with system peripherals. FLD is the routine which acts as a system executive and links the service modules to perform the desired data gathering task. Functional descriptions of FLD and the various service modules are given below.

1. FLD

FLD is the executive program for the sequential strain monitor system. FLD calls LGAP to prepare the tape for recording and then writes the header information (time, date and number of channels to be monitored) to facilitate data identification.

FLD then initializes to zero the reference vectors which will be used to store the last incremental strain level, XLST, and the change in strain level, DXLST, initializes the pointers for the output buffer and begins the monitor task. The DAS-16 is operated in the sequential mode and short-counted in software using information from the header concerning the top channel number to be monitored.

When the flag set by FLTR (see description below) is sensed, FLD stores the data points in the output buffer and continues to monitor until the buffer is full or the record portion of the program is activated by the user through the JUMP routine.

Figures 2 and 3 show the output from the FLD and FLTR combination. Figure 2 shows a comparison of the continuously monitored signals, a five Hz. sine wave and a zero reference signal, and the "filtered" data extracted from them. This output was obtained without the flagging function mentioned above enabled in order to demonstrate that only reversal points are selected for recording. When

the flag function is enabled, the output from the same input signals appears as in figure 3, where only the "filtered" data points are recorded. A listing of FLD is contained in the sequential strain monitor software listing.

2. FLTR

The FLTR routine monitors the incoming strain data to determine if a fatigue-significant event has occurred. The two reference vectors, XLST and DXLST, mentioned above are used to maintain a running account of incremental strain level and the change in that level for each channel. XLST and DXLST are updated for each five percent change in strain level and when the proper criteria have been met (a five percent or better change from the last point and a reversal of algebraic sign) a flag is set and all channels will be stored in the output buffer. A listing of FLTR is contained in the sequential strain monitor software listing.

3. Service Modules

The service modules described below are a part of the software developed during the design and construction phase of the MIDAS/Sequential Strain Monitor system. The MIDAS software listings contain some routines not described here but they are not used in the strain monitor routines.

a. JUMP

JUMP allows the user to enter the address of a routine which he wishes to exercise, then jumps to that address and executes the routine. This is the technique used for accessing FLD.

To use JUMP, the user resets the system using the reset button; then enters the line number and page number of the routine he wishes to access. The description of KEY has more details on keyboard operation.

b. LGAP

LGAP writes a long unmarked leader on the magnetic tape to ensure that no information is inadvertently lost or garbled by attempting to record on the clear leader or the first, possibly soiled, few inches of tape. LGAP is used in all routines which write to the recorder.

c. SGAP

SGAP writes a four byte long, unmarked gap between records to ensure that the recorder is allowed sufficient space to react to start/stop commands without overwriting information while recording or causing "bit dropout" during read operations. Like LGAP, SGAP is used in all routines which transfer information to the magnetic tape.

d. CLK

CLK activates the DISC (Digital Input Select Card) which controls the input to Port 0 of the microprocessor. The current hour and minute reading from the digital clock is then stored in memory at the location indicated before the call. CLK is used by all data routines to put an identifying time in the header information.

e. KEY

KEY is the routine which reads the information entered via the keyboard and stores it in the memory location set before the call. KEY is used to enter the header information needed to short-count the DAS-16 and to identify each data run.

KEY stores the information entered in register A and displays its contents to the user through the hexadecimal displays. The data are not accepted for use until the "Word Entry" key is depressed, allowing the user to correct any errors in keying before entering the data.

f. RCDR

RCDR is the routine which controls the tape recorder and feeds it the information from the output buffer. The motor clock and write clock are both generated in software by this routine, allowing absolute control over the recorder. The parameters which must be supplied to the routine are: the beginning address of the data to be recorded, the number of bytes to be recorded and whether the operation is to be a write or erase. The maximum number of data words which may be recorded during one call to RCDR is 256, the contents of one "page" of memory.

H. DATA EXTRACTION

To facilitate data extraction, a separate reader was constructed and interfaced to a Hewlett Packard HP-9830 programmable calculator. The reader consists of a read-write tape recorder, identical to that in the sequential strain monitor system, coupled to a Pro-Log Corporation PL-803 microcomputer. The PL-803 is very similar to the microcomputer used in the sequential strain monitor/MIDAS system but consists of only three printed circuit cards and has only 28 input/output lines instead of the 64 for the larger system. The HP-9830 controls the PL-803 which, in turn, controls the recorder to extract the data and store it in the HP-9830 memory. Two options exist for data display; a graphical presentation such as shown in figures 2 and 3, or a tabular output such as figure 5. The program for the HP-9830 is written in BASIC and is listed in the software section, as is the assembly language program listing for the PL-803.

I. INTERFACE WIRING TABLES

The tables on the following pages are provided in lieu of complex wiring diagrams. It is felt that this format is easier to comprehend and more useful for duplication of the system if desired. The asterisk notation in the tables indicates the logical NOT operation.

Table I. KEY TO SYMBOLS USED IN INTERCONNECTION TABLES

SYMBOL	MEANING
Ixx	Input card, pin xx
Oxx	Output card, pin xx
Kx	Keyboard,pin x
D-jn-xx	DAS-16, jack n, pin xx
Mxx	Memodyne, pin xx
Cxx	Clock, pin xx
DIxx	DISC pin xx
DI-n-x	DISC input port n, pin x
Hn-xx	Hexidecimal display number n, pin xx

Table II. INPUT PORT 0-KEYBOARD

BIT	I/O TERM	DIST TERM	LINE NAME
8	I-41	K-0	Keyboard 0*
7	I-43	K-W	Keyboard word entry*
6			no connection
5			no connection
4	I-49	K-8	Keyboard 8*
3	I-51	K-4	Keyboard 4*
2	I-53	K-2	Keyboard 2*
1	I-55	K-1	Keyboard 1*

Table III. INPUT PORT 0-DISC

BIT	I/O TERM	DIST TERM	LINE NAME
8	I-41	DI-13	DISC digital out 8
7	I-43	DI-12	DISC digital out 7
6	I-45	DI-16	DISC digital out 6
5	I-47	DI-18	DISC digital out 5
4	I-49	DI-19	DISC digital out 4
3	I-51	DI-17	DISC digital out 3
2	I-53	DI-15	DISC digital out 2
1	I-55	DI-14	DISC digital out 1

Table IV. INPUT PORT 1-MEMODYNE/DAS-16

BIT	I/O TERM	DIST TERM	LINE NAME
8	I-42	M-7	Tape clock
7	I-44	M-6	Tape data
6			no connection
5			no connection
4	I-50	D-j2-D	DAS-16 digital out 9
3	I-52	D-j2-C	DAS-16 digital out 10
2	I-54	D-j2-B	DAS-16 digital out 11
1	I-56	D-j2-A	DAS-16 digital out 12(LSB)

Table V. INPUT PORT 2-DAS-16

BIT	I/O TERM	DIST TERM	LINE NAME
8	I- 8	D-j2-N	DAS-16 digital out 1(MSB)
7	I-10	D-j2-M	DAS-16 digital out 2
6	I-12	D-j2-L	DAS-16 digital out 3
5	I-14	D-j2-K	DAS-16 digital out 4
4	I-16	D-j2-J	DAS-16 digital out 5
3	I-18	D-j2-H	DAS-16 digital out 6
2	I-20	D-j2-H	DAS-16 digital out 7
1	I-22	D-j2-E	DAS-16 digital out 8

Table VI. INPUT PORT 3-DAS-16

BIT	I/O TERM	DIST TERM	LINE NAME
8	I-7	D-j2-R	DAS-16 Busy/End-of-convert*
7			no connection
6			no connection
5			no connection
4			no connection
3			no connection
3			no connection
2			no connection
1			no connection

Table VII. OUTPUT PORT 0-DISPLAY 0

BIT	I/O TERM	DIST TERM	LINE NAME
8	0-42	H0-80	hexadecimal display 0, 80
7	0-44	H0-40	hexadecimal display 0, 40
6	0-46	H0-20	hexadecimal display 0, 20
5	0-48	H0-10	hexadecimal display 0, 10
4	0-50	H0-08	hexadecimal display 0, 08
3	0-52	H0-04	hexadecimal display 0, 04
2	0-54	H0-02	hexadecimal display 0, 02
1	0-56	H0-01	hexadecimal display 0, 01

Table VIII. OUTPUT PORT 1-DISPLAY 1

BIT	I/O TERM	DIST TERM	LINE NAME
8	0-41	H1-80	hexadecimal display 1, 80
7	0-43	H1-40	hexadecimal display 1, 40
6	0-45	H1-20	hexadecimal display 1, 20
5	0-47	H1-10	hexadecimal display 1, 10
4	0-49	H1-08	hexadecimal display 1, 08
3	0-51	H1-04	hexadecimal display 1, 04
1	0-55	H1-01	hexadecimal display 1, 01:

Table IX. OUTPUT PORT 2-DAS-16/DISC

BIT	I/O TERM	DIST TERM	LINE NAME
8	O- 8	D-j1-13	DAS-16 Random Address 8*
7	O-10	D-j1-12	DAS-16 Random Address 4*
6	O-12	D-j1-N	DAS-16 Random Address 2*
5	O-14	D-j1-P	DAS-16 Random Address 1*
4	O-16	DI-Y	DISC PORT 1 Select
4	O-16	DI-X	DISC PORT 2 Select
2	O-20	DI-C	DISC PORT 3 Select
1	O-22	DI-B	DISC PORT 4 Select

Table X. OUTPUT PORT 3-DAS-16/MEMODYNE

BIT	I/O TERM	DIST TERM	LINE NAME
8	O-7	D-j1-11	DAS-16 RESET*
7	O-9	D-j1-9	DAS-16 CONVERT*
7	O-11	D-j1-10	DAS-16 RANDOM/SEQUENTIAL*
5	O-13	D-j1-M	DAS-16 STROBE*
4	O-15	M-22	MEMODYNE WRITE/READ*
3	O-17	M-1	MEMODYNE WRITE CLOCK
2	O-19	M-10	MEMODYNE MOTOR CLOCK
1	O-21	M-2	MEMODYNE SERIAL DATA IN

Table XI. CLOCK-DISC

CLOCK	DISC	USE
C-m-1	DI-1-8	unit minutes digit select
C-m-10	DI-1-7	ten minutes digit select
C-h-1	DI-1-6	unit hours select
C-h-10	DI-1-5	ten hours select
C-8	DI-1-4	BCD 8
C-4	DI-1-3	BCD 4
C-2	DI-1-2	BCD 2
C-1	DI-1-1	BCD 1

Table XII. POWER DISTRIBUTION

TERM	POWER	USE
I-1,2	+5v, 2a.	microcomputer
I-3,4	0	microcomputer ground
I-5,6	-10v, 1a.	microcomputer
M-13	+5v, 1.3a.	recorder
M-14	0	recorder ground
M-15	+12v, 1a.	recorder
D-j2-Y,Z	+5v, 0.8a.	DAS-16
D-j2-V,S,13	-15v,0.07a.	DAS-16
D-j2-W,X	0	DAS-16 ground
D-j2-U	+15v,0.13a.	DAS-16
D-j1-L	0	DAS-16 device select
D-j2-22	0	DAS-16 sequencer enable
DI-1	+5v, 0.2a.	DISC
DI-22	0	DISC ground
C-+	+5v, 0.2a.	clock
C--	-10v,0.1a.	clock
H-+	+5v, 0.8a.	display
H--	0	display ground

VI. SUGGESTIONS FOR FUTURE RESEARCH

Several areas exist where future research or investigation could prove very useful. There is an immediate need for a follow-on prototype of the fatigue life data system which should see flight test in Naval aircraft. From the author's experience with the first prototype, several features of the follow-on prototype have suggested themselves. It should not be necessary to have the large input/output capabilities associated with a general data system such as MIDAS; a smaller microcomputer, such as the PL-803 described earlier, could be used. An eight bit analog-digital converter could profitably be employed in place of the twelve bit model modified for the MIDAS system. There should be no necessity for interaction with the program; the keyboard could be omitted and function switches installed to perform the header writing and output buffer dump now done with the keyboard. A write-only tape deck could be used in place of the more expensive read-write model used for MIDAS, but some ground-based recorder facility for extraction of data would be needed to support such a system. A single unit cost for the flight system components has been estimated at $2000.00 for a system occupying about one cubic foot and weighing approximately three pounds.

Automation of data extraction and handling, including the preparation of magnetic tape data records directly compatible with existing computer hardware and software at NADC, is possible with this system in the short-term future. A system for transcribing data cassettes directly to IBM or CDC compatible tape at the squadron/ship/air station level would be straightforward and would do much to alleviate the problems of data handling already discussed.

VII. CONCLUSIONS

Future fatigue life monitoring/prediction programs will of necessity require more accurate data and more sophisticated damage calculation techniques to permit more efficient utilization of weapons systems by safely prolonging their service lives. It is believed that the multichannel, sequential strain monitor system proposed herein affords at least a part of the answer to meeting those requirements in the not-too-distant future. The coupling of its proven ability to extract fatigue-significant data from a continuously monitored loading sequence with an updated damage calculation program would be a credible first step toward that goal.

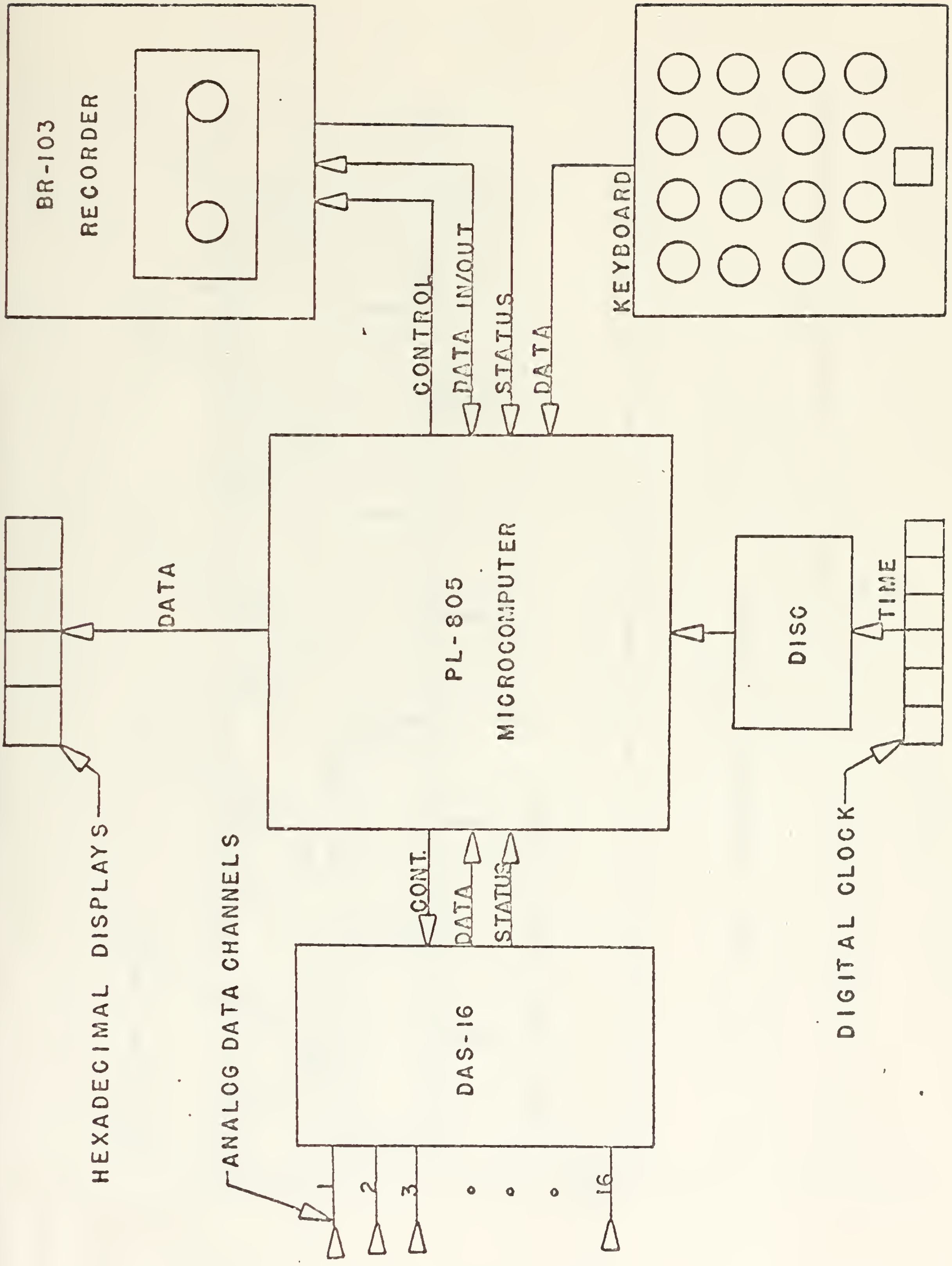

BR-103
RECORDER
CONTROL
DATA IN/OUT
STATUS
DATA
KEYBOARD
PL-805
MICROCOMPUTER
DATA
HEXADECIMAL DISPLAYS
DISC
TIME
DIGITAL CLOCK
CONT.
DATA
STATUS
DAS-16
ANALOG DATA CHANNELS
1
2
3
16

Figure 1

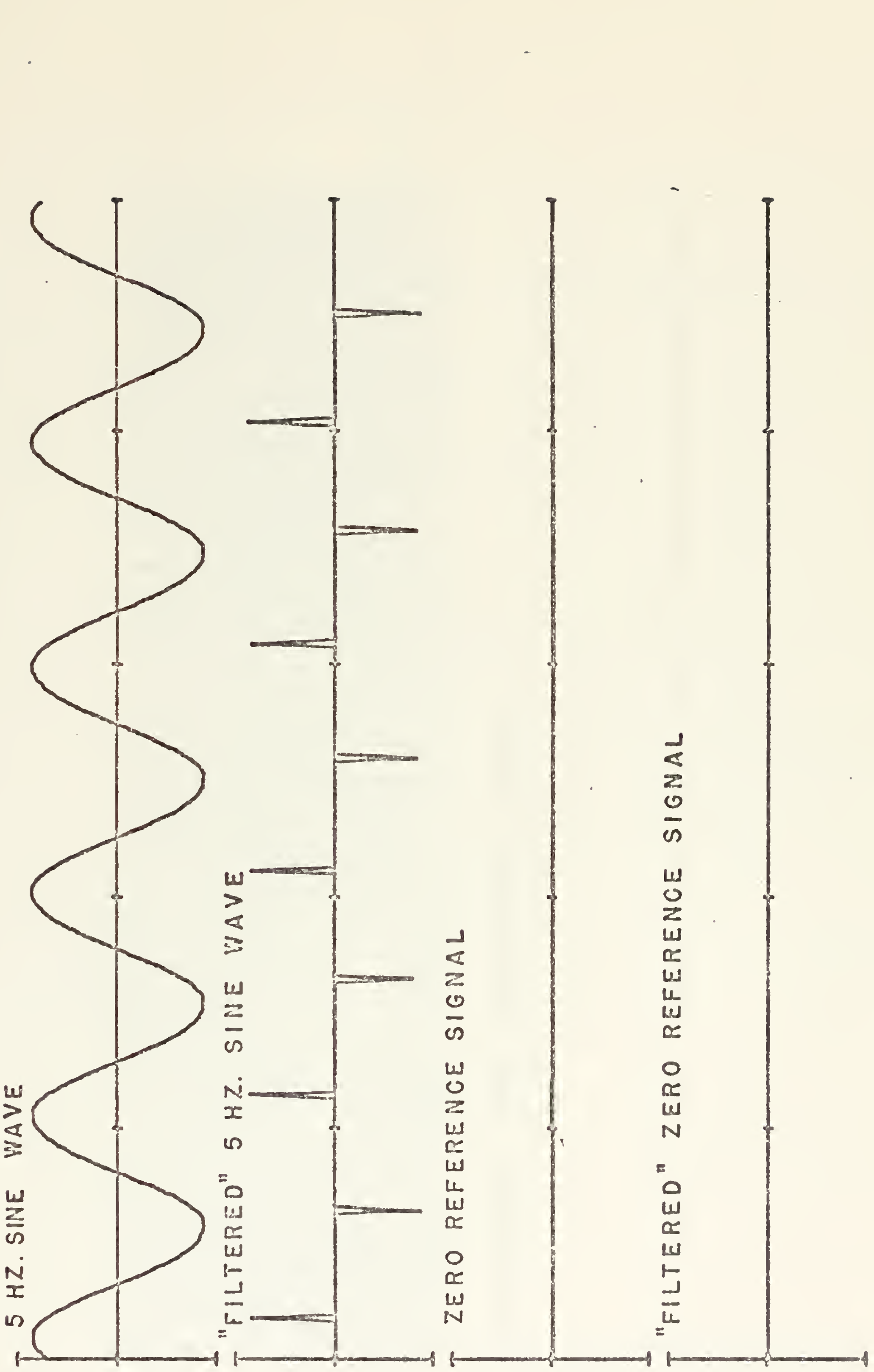

5 HZ. SINE WAVE
"FILTERED" 5 HZ. SINE WAVE
ZERO REFERENCE SIGNAL
"FILTERED" ZERO REFERENCE SIGNAL

Figure 2

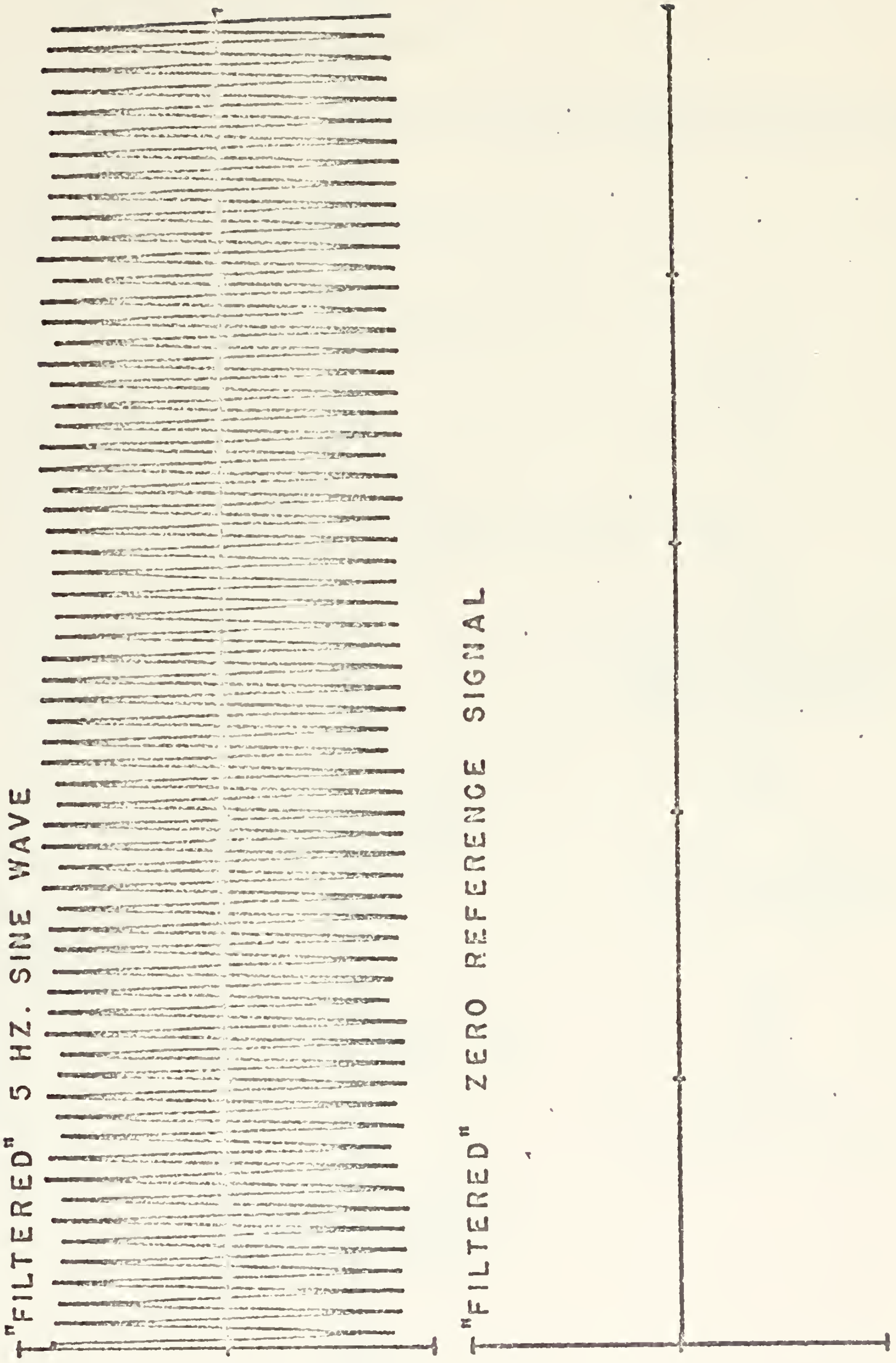


Figure 3

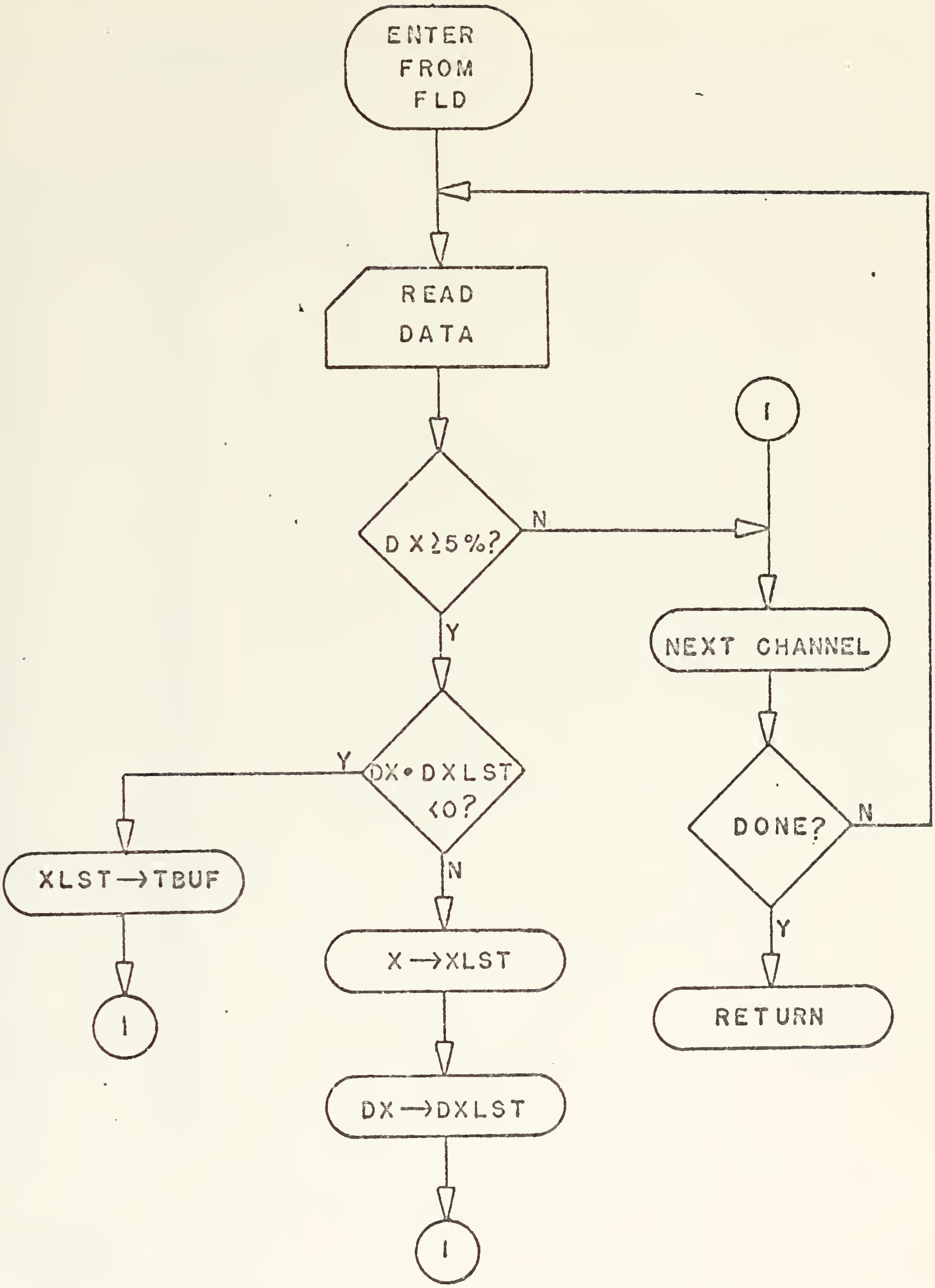

ENTER FROM FLD
READ DATA
DX≥5%?
N
Y
DX•DXLST <0?
Y
N
XLST→TBUF
1
X→XLST
DX→DXLST
1
1
NEXT CHANNEL
DONE?
N
Y
RETURN

Figure 4

```
 FATIGUE LIFE STRAIN ROUTINE OUTPUT

      OPTIONS AVAILABLE
      1. TABULAR OUTPUT
      2. PLOTTER OUTPUT



HEADER INFORMATION
TIME 0933    DATE 05 14 75
TOP CHANNEL IN CYCLE 02
CHAN 1     CHAN 2
 19        128
237        128
 20        128
236        128
 20        128
237        128
 19        128
237        128
 18        128
238        128
 18        128
237        128
 19        128
237        128
 20        128
236        128
 20        128
237        128
 19        128
237        128
 18        128
238        128
 18        128
237        128
 19        128
237        128
 20        128
236        128
 20        128
237        128
 19        128
237        128
  .          .
  .          .
  .          .
```

Figure 5

THE
DESIGNERS GUIDE
TO PROGRAMMED LOGIC

For MPS 800 Systems

Written By

MATT BIEWER

Reprinted with permission of

PRO-LOG CORPORATION
852 Airport Road
Monterey, California

1. MPS 800 SUMMARY

MPS 800 systems are 8 bit microcomputer card systems suitable for implementing random logic and alpha-numeric data handling applications. As shown in Figure 1-1, the MPS 800 system consists of a CPU card, ROM and RAM memory cards and I/O cards. The ROM memory cards are implemented to accept either programmable erasable PROMs for speedy program implementation or pin compatible masked ROMs for volume production. The CPU card uses the 8008 microprocessor chip.

CPU CAPABILITIES

- Fourteen bit program address
- Seven level address stack for subroutines
- Six, eight bit general purpose registers
- Eight bit accumulator plus carry
- Arithmetic and accumulator instructions
    - Add and subtract with or without carry
    - Rotate left or right through or around carry
- Logical instructions
    - AND, OR, Exclusive OR, COMPARE
- Decision making (address control instructions)
    - Test operations for zero or nonzero result
    - Test carry for logic one or zero
    - Test operation parity for odd or even result
    - Test operation sign (MSB) for logic one or zero result
- Input/output instructions with direct addressing
- Memory instructions
    - Register to Memory
    - Memory to register
    - Logical from memory
    - Arithmetic from memory
- Register instructions
    - Register to register load
    - Increment and Decrement
- Expandable single line interrupt
- CPU disconnects from Data, address, and control for DMA

MEMORY

- ROM and RAM, program or data storage up to 16,384 words
- Programmable erasable ROMs: 5202, 1702 type

INPUT/OUTPUT

- 192 TTL output lines directly selectable
- 64 TTL input lines directly selectable
- Input instruction, gates data into the CPU accumulator
- Output instruction, latches accumulator data at output

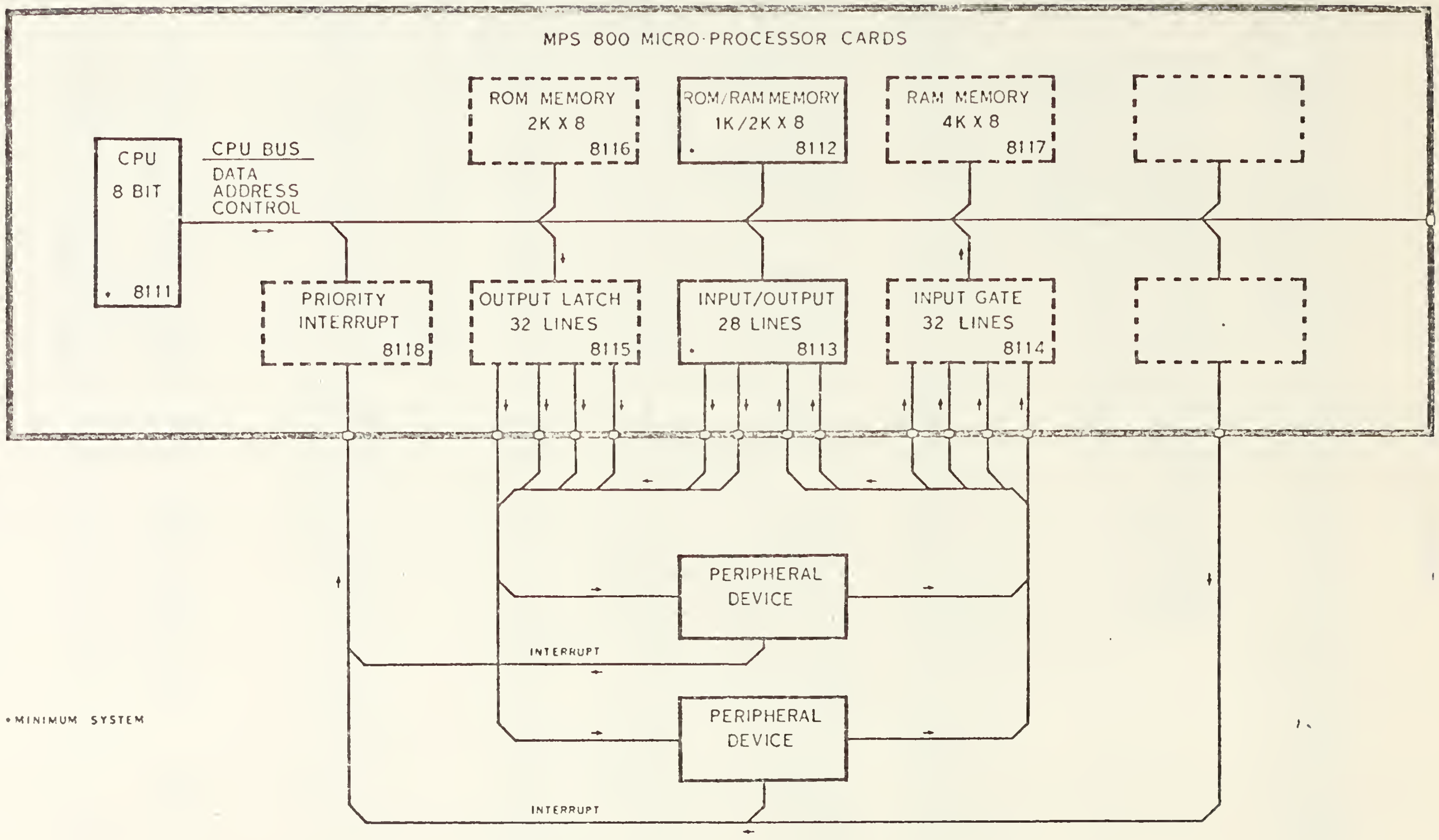


Figure 1-1 MPS 800 Microprocessor System

2. MPS 800 HARDWARE

The MPS 800 series provides a choice of microprocessor card sets with differing expandability. Each set is complete with CPU, ROM memory, RAM memory, input and output. All sets are assembled using combinations of the following cards:

8111	CPU with data, address, and control
8112	ROM/RAM combination 1K ROM, 2K RAM
8113	I/O combination of 28 TTL I/O lines
8114	INPUT 32 gated inputs
8115	OUTPUT 32 latched outputs
8116	ROM 2K memory
8117	RAM 4K memory

The MPS 803 is complete on three cards providing the lowest cost combination for limited system size. It includes the 8111 CPU, 8112 ROM/RAM and 8113 I/O. Direct system expansion is limited to 4096 words of ROM and 8192 words of RAM and a combination of 28 I/O lines.

The MPS 805 is complete on five cards providing maximum expansion capability of ROM, RAM, and I/O. It includes the 8111 CPU, 8114 Input, 8115 Output, 8116 ROM, and 8117 RAM. Expansion is unlimited to the full address capability of 16,384 words of memory in any combination of RAM and ROM, 192 output lines, and 64 input lines.

The user may assemble other sets based on his need. A system requiring expandable I/O with minimum memory requirements can be assembled on four cards using the 8111 CPU, 8112 ROM/RAM, 8114 Input and the 8115 Output.

A system requiring expandable memory with minimum I/O requirements can be assembled on four cards using the 8111 CPU, 8113 I/O, 8116 ROM and the 8117 RAM.

INTERCONNECTING MPS 800 SYSTEMS

The partitioning of the MPS 800 system provides a simple interconnect scheme. Most connections are made from the 8111 CPU to the other system elements. The 8111 interface consists of a data bus, address bus and control bus.

DATA BUS

DINx*	Data in inputs. The 8 bit data input bus. Data from RAM, ROM, INPUT, Interrupt, etc: to the CPU.
DOUT*	Data out outputs. The 8 bit data output bus. Data from the CPU to RAM memory.

ESS BUS

Address outputs. The 14 line address bus. Address to ROM and RAM memory. Port selection to I/O. Output data for output ports.

ROL BUS

	Signal from the CPU for gating input port data to the CPU.
	Signal from the CPU for strobing output port data from the CPU to the output latches.
	Write memory output. Signal for strobing memory write data from the CPU to RAM memory.
	Read memory output. Signal for gating memory read data to the CPU.
*	Read memory interrupt output. If the system is wired for interrupt, this signal is used instead of RDM to gate memory read data to the CPU. RDMI* does not occur during interrupt cycles.
	Memory ready input. This signal causes the CPU to wait for input data on the DIN bus or holds output data on the DOUT bus.
*	Interrupt response output. A signal which occurs in response to the IREQ* input. INTR* is used externally to gate interrupt data to the DIN bus.
*	Interrupt request input. A signal on this line causes the CPU to execute an interrupt cycle.
*	Interrupt hold input. A signal on this line during interrupt causes INTR* to remain active as long as IHLD* is true. This line allows multi-byte interrupt.
	Restart input. A signal on this line during reset prevents power-on interrupt.
T*	Reset input. A signal on this line causes the program address counter in the CPU to be reset to zero.
	Reset output. A signal which occurs in response to the RESET* input or power turn on. This signal can be used to reset the output latches. If RST* is wired to RS*, power-on restart is disabled. If RST* is wired IREQ*, power-on restart is enabled.
*	STOP* indicates that the CPU is halted and all address, data, and control lines are released by the CPU for DMA operations.

MPS 803 8 BIT MICROPROCESSOR SYSTEM

A three card 8 bit microcomputer which implements the 8008 CPU into a working system with ROM/RAM and I/O. The system includes features to accommodate DMA and Interrupt or power-on external restart. The 803 is designed for low cost control or minimum data handling applications.

FEATURES

- 8008 CPU with Interrupt and DMA capability
- All memory usable as data or program storage
- 256 words of ROM memory with 1024 word capacity
- 1024 words of RAM memory with 2048 word capacity
- 28 TTL I/O lines field selectable as input or output
- Limited expandability on ROM, RAM and I/O

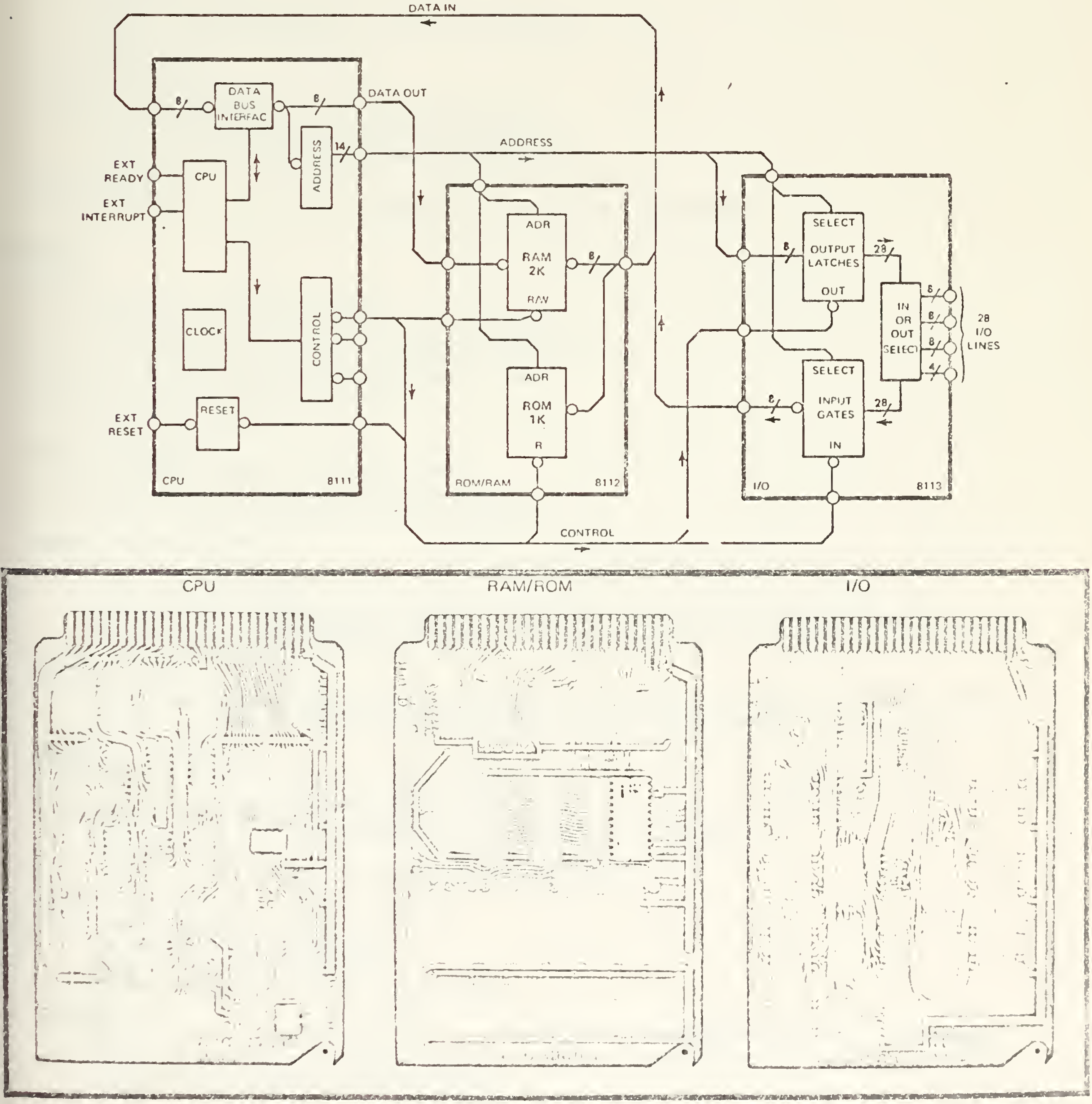


Figure 2-1 MPS 803 8 Bit Microprocessor

MPS 803 MICROPROCESSOR SYSTEM SPECIFICATIONS

Physical

Three 4.5" by 6.5" printed circuit cards
- One 8111 CPU card
- One 8112 ROM/RAM card
- One 8113 I/O card

Connector Requirements for each card
56 pin, 28 position dual read-out on 0.125 centers

CPU Card includes
8008 CPU
Crystal clock
Address latches, data buffers, and control decode circuits.
Power-on and external restart.
DMA buffers.

ROM/RAM card includes
One 1702A PROM (256 bytes) and four PROM sockets
Eight 2102 RAM (1024 bytes) and sixteen RAM sockets
Socket for card expansion circuit. Limit 2 ROM/RAM cards

I/O card includes
28 TTL I/O circuits selectable in groups of 4 as input gates or output latches.

Operational

CPU
Executes all of the 8008 instructions.
4 microsecond time state cycle using 8008 (MPS 803).
2.8 microsecond time state cycle using 8008-1 (MPS 803-1).

Memory for data or program storage expandable to 2 cards
ROM, 1024 word capacity per card.
RAM, 2048 word capacity per card.

Input and Output
Input gates implement the INP instructions.
Output latches implement the OUT instructions.
Requires external address decoding for card expansion.

Interrupt or External Restart
Single line, synchronized interrupt on CPU card can be optionally wired for multi-level interrupt or Power-on external restart.
Multi-level Interrupt: Control lines available for external interrupt such as 8118 priority interrupt card.
Power-on and external restart option: CPU starts at instruction location 0000 by wiring restart output from CPU card to Interrupt Request input.

DMA (Direct Memory Access)
Data, address, and control lines are 3-state disconnected by the CPU following a HLT instruction allowing DMA by peripherals. The CPU must be interrupted to continue following a HALT.

Electrical Requirements
Refer to individual data sheets and schematics on the 8111, 8112, and 8113 for interface and wiring.

Power Requirements for the three card set fully loaded
+VCC = +5 volts ±5% @ 2 Amp maximum (35mA per ROM, 50 mA per RAM)
GND 0 volts
−VDD = −9 volts ±5% @ 750 mA maximum (35 mA per ROM)

Hardware Accessories
Compatible with Series 8400 interface cards.
Fits CR5, CR10 or CR19 card racks
Use M272 or M273 power supply
PROM's programmable on Series 81 programmers

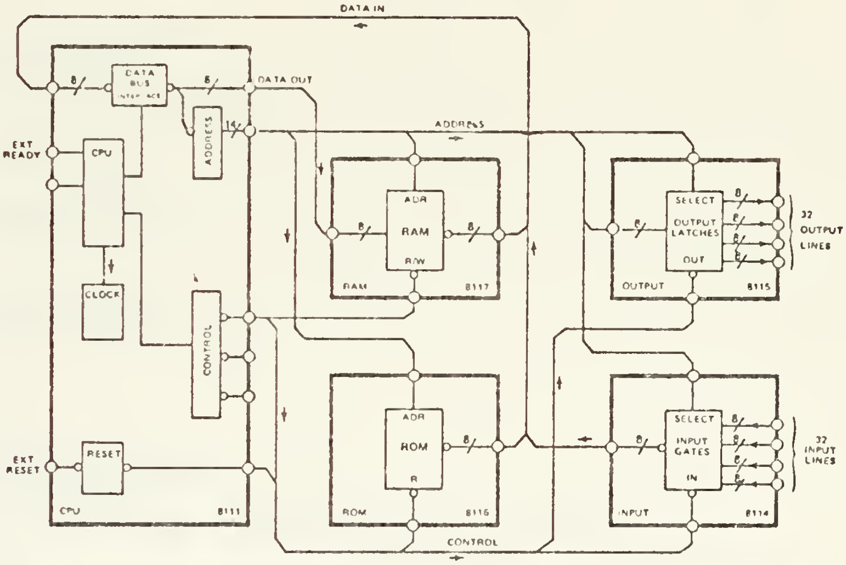
Software
MPS 803 hardware is fully compatible with any 8008 software assuming I/O and interrupt can be assigned compatibly. Teletype operating system and system monitor available. Assemblers, compilers and simulators available through computer time-sharing services.

MPS 805 8 BIT MICROPROCESSOR SYSTEM

A five card 8 bit microcomputer which implements the 8008 CPU into a working system with ROM, RAM, Input and Output. The system includes features to accommodate DMA and Interrupt or power-on external restart. The 805 is designed as a fully expandable system for programmed logic applications, moderate data handling, or minicomputer replacement.

FEATURES:

- 8008 CPU with Interrupt and DMA capability.
- Crystal clock with better than 0.1% accuracy.
- All memory usable as data or program storage.
- 256 words PROM memory with 2048 word capacity.
- 1024 words RAM memory with 4096 word capacity.
- ROM and RAM memory card expandable to a total of 16K words.
- 32 TTL output latches card expandable to 192 latches.
- 32 TTL input gates card expandable to 64 inputs.



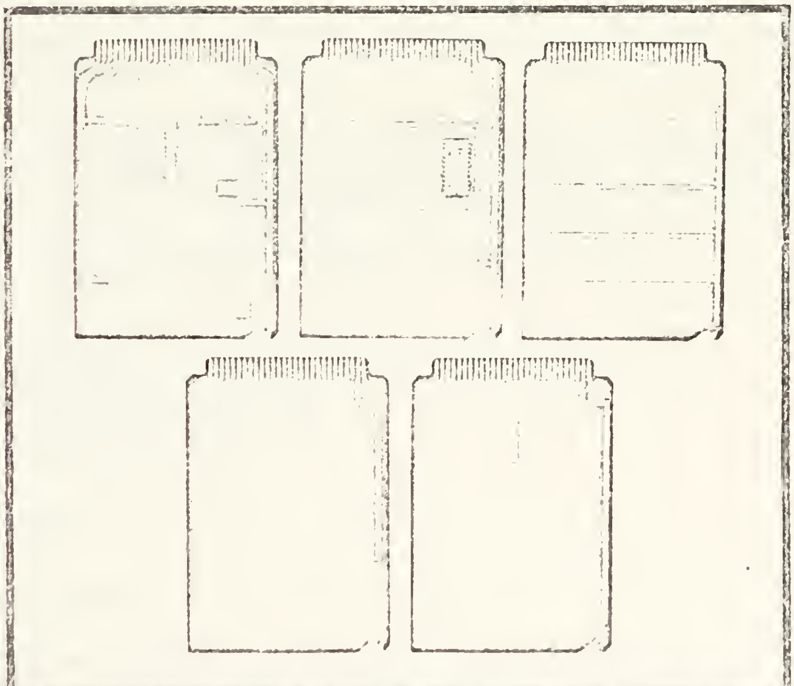


Figure 2-2 MPS 805 8 Bit Microprocessor

MPS 805 MICROPROCESSOR SYSTEM SPECIFICATIONS

Physical

Three 4.5" by 6.5" printed circuit cards
- One 8111 CPU card
- One 8114 Input card
- One 8115 Output card
- One 8116 ROM card
- One 8117 RAM card

Connector Requirement for each card
56 pin, 28 position dual read-out on 0.125 centers

CPU Card includes
8008 CPU
Crystal clock
Address latches, data buffers, and control decode circuits.
Power-on and external restart.
DMA buffers.

ROM Card includes
One 1702A PROM (256 bytes) and eight PROM sockets
Socket for card expansion circuit (up to 8 cards)

RAM Card includes
Eight 2102 RAM (1024 bytes) and thirty-two RAM sockets
Socket for card expansion circuit (up to 4 cards)

Input Card includes
32 TTL input selector circuits addressable in groups of 8
Socket for card expansion circuit (up to 2 cards)

Output Card includes
32 TTL output latch circuits addressable in groups of 8
Socket for card expansion circuit (up to 6 cards)

Operational

CPU
Executes all of the 8008 instructions.
4 microsecond time state cycle using 8008 (MPS 1805).
2.8 microsecond time state cycle using 8008-1 (MPS 805-1).

Memory for data or program storage card expandable to any combination of ROM and RAM to 16384 words
ROM, 2048 word capacity per card.
RAM, 4096 word capacity per card.

Input and Output
Input gates implement the INP instructions.
Output latches implement the OUT instructions.

Interrupt or External Restart
Single line, synchronized interrupt on CPU card can be optionally wired for multi-level interrupt or Power-on external restart.
Multi-level Interrupt: Control lines available for external interrupt such as 8118 priority interrupt card.
Power-on and external restart option: CPU starts at instruction location 0000 by wiring restart output from CPU card to Interrupt Request input.

DMA (Direct Memory Access)
Data, address, and control lines are 3-state disconnected by the CPU following a HLT instruction allowing DMA by peripherals. The CPU must be interrupted to continue following a HALT.

Electrical Requirements
Refer to individual data sheets and schematics on the 8111, 8114, 8115, 8116, and 8117 for interface and wiring.

Power Requirements for the five card set fully loaded
+VCC = ±5% @ 3.3 Amp maximum (35mA per ROM, 50mA per RAM)
GND 0 volts
−VDD = −9 volts ±5% @ 900 mA maximum (35 mA per ROM)

Hardware
Compatible with Series 8400 interface cards.
Fits CR5, CR10 or CR19 card racks
Use M273 power supply
PROM's programmable on Series 81 programmers

Software
MPS 800 hardware is fully compatible with any 8008 software assuming I/O and interrupt can be assigned compatibly. Teletype operating system and system monitor available. Assemblers, compilers and simulators available through computer time-sharing services.

3. MPS 800 TIMING

INSTRUCTION CYCLE

The MPS 800 systems operate as do all computers by periodically and continuously retrieving groups of bits from a program memory and performing operations defined by the bit patterns of these instruction words. The instruction cycle of addressing the program memory, fetching the instruction word and executing the operation requires multiple time intervals. The total time to complete the instruction cycle is called the instruction cycle time. The instruction cycle time for the MPS 800 systems is variable, requiring more time to execute some instructions than others.

TIME STATES

The instruction cycle times for various types of instructions fit into fifteen time intervals or time states as shown in Figure 3-1. In each time state, the CPU operates with 8 bits of information. The instructions may use as few as three time states or as many as eleven. Each time state is defined by the ϕ_1 and ϕ_2 clock signals and the SYNC signal. The frequency of the two-phase clock determines the actual time of the time states. The MPS 800 operates with a 4.0 microsecond time state and the MPS 800-1 operates with a 2.8 microsecond time state.

MEMORY CYCLES

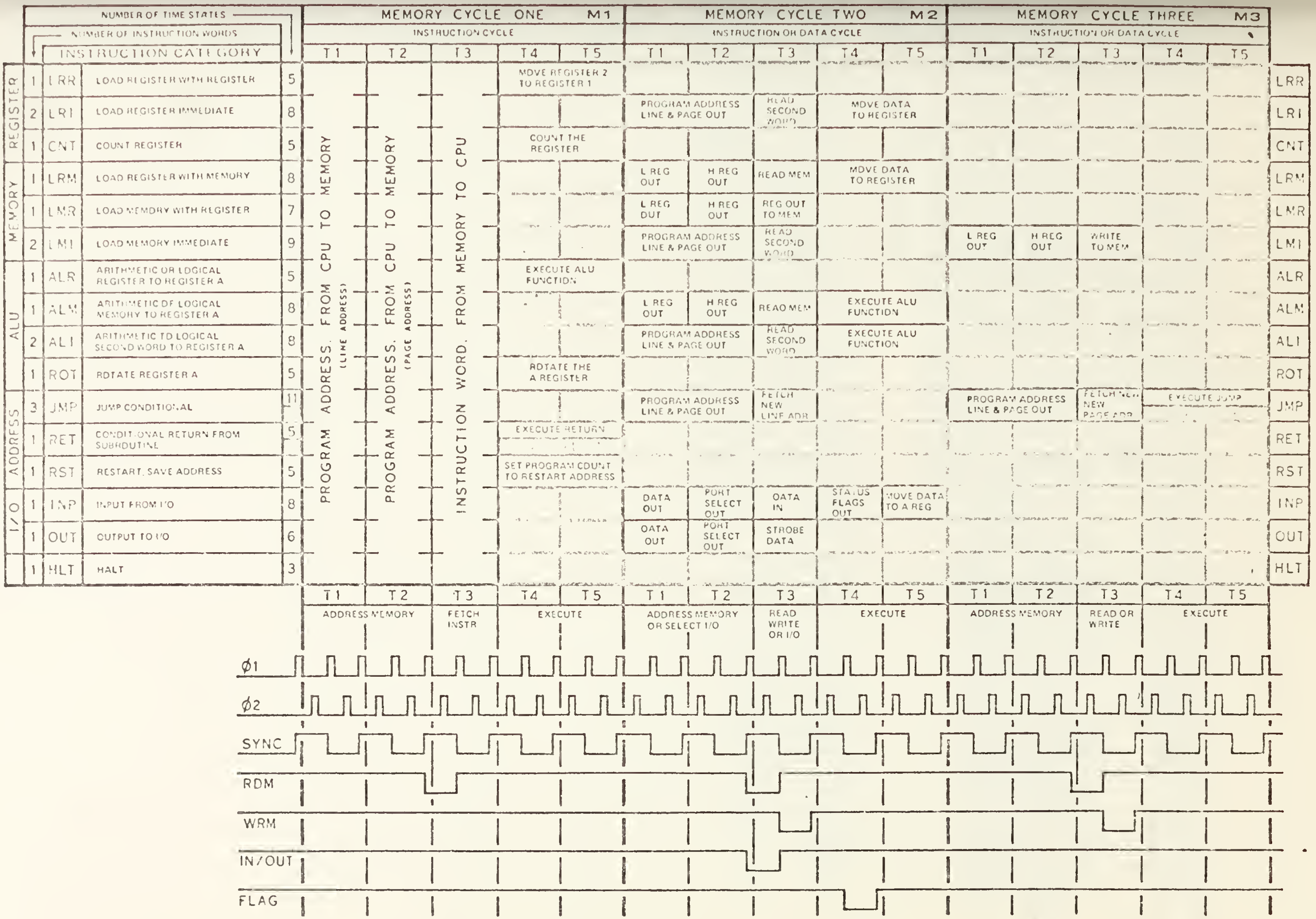
The 15 time intervals are divided into three memory cycles, each with 5 time states. For each memory cycle the first two time states T_1 and T_2 are used for addressing either memory or I/O. The third time state T_3 is always information retrieval or storage. The last two time states T_4 and T_5 are execution time states which do not occur every memory cycle.

The instruction set includes single, double and triple word instructions which require one, two or three memory cycles. The instruction set also includes various single word instructions which require two memory cycles and a double word instruction requiring three memory cycles.

INSTRUCTION OR DATA CYCLE

A memory cycle can be an instruction cycle or a data cycle depending on whether memory is addressed by the program address counter or the memory registers within the CPU.

The first memory cycle M1 is always an instruction cycle where the program address counter addresses the program memory and the instruction is fetched to the instruction register. The second and third memory cycles, M2 and M3, can be either instruction or data cycles.



	NUMBER OF INSTRUCTION WORDS		INSTRUCTION CATEGORY	NUMBER OF TIME STATES	MEMORY CYCLE ONE M1 INSTRUCTION CYCLE T1	T2	T3	T4	T5	MEMORY CYCLE TWO M2 INSTRUCTION OR DATA CYCLE T1	T2	T3	T4	T5	MEMORY CYCLE THREE M3 INSTRUCTION OR DATA CYCLE T1	T2	T3	T4	T5	
REGISTER	1	LRR	LOAD REGISTER WITH REGISTER	5	PROGRAM ADDRESS. FROM CPU TO MEMORY (LINE ADDRESS)	PROGRAM ADDRESS. FROM CPU TO MEMORY (PAGE ADDRESS)	INSTRUCTION WORD. FROM MEMORY TO CPU	MOVE REGISTER 2 TO REGISTER 1												LRR
	2	LRI	LOAD REGISTER IMMEDIATE	8						PROGRAM ADDRESS LINE & PAGE OUT		READ SECOND WORD	MOVE DATA TO REGISTER							LRI
	1	CNT	COUNT REGISTER	5				COUNT THE REGISTER												CNT
MEMORY	1	LRM	LOAD REGISTER WITH MEMORY	8						L REG OUT	H REG OUT	READ MEM	MOVE DATA TO REGISTER							LRM
	1	LMR	LOAD MEMORY WITH REGISTER	7						L REG OUT	H REG OUT	REG OUT TO MEM								LMR
	2	LMI	LOAD MEMORY IMMEDIATE	9						PROGRAM ADDRESS LINE & PAGE OUT		READ SECOND WORD			L REG OUT	H REG OUT	WRITE TO MEM			LMI
ALU	1	ALR	ARITHMETIC OR LOGICAL REGISTER TO REGISTER A	5				EXECUTE ALU FUNCTION												ALR
	1	ALM	ARITHMETIC OR LOGICAL MEMORY TO REGISTER A	8						L REG OUT	H REG OUT	READ MEM	EXECUTE ALU FUNCTION							ALM
	2	ALI	ARITHMETIC TO LOGICAL SECOND WORD TO REGISTER A	8						PROGRAM ADDRESS LINE & PAGE OUT		READ SECOND WORD	EXECUTE ALU FUNCTION							ALI
	1	ROT	ROTATE REGISTER A	5				ROTATE THE A REGISTER												ROT
ADDRESS	3	JMP	JUMP CONDITIONAL	11						PROGRAM ADDRESS LINE & PAGE OUT		FETCH NEW LINE ADR			PROGRAM ADDRESS LINE & PAGE OUT		FETCH NEW PAGE ADR	EXECUTE JUMP		JMP
	1	RET	CONDITIONAL RETURN FROM SUBROUTINE	5				EXECUTE RETURN												RET
	1	RST	RESTART, SAVE ADDRESS	5				SET PROGRAM COUNT TO RESTART ADDRESS												RST
I/O	1	INP	INPUT FROM I/O	8						DATA OUT	PORT SELECT OUT	DATA IN	STATUS FLAGS OUT	MOVE DATA TO A REG						INP
	1	OUT	OUTPUT TO I/O	6						DATA OUT	PORT SELECT OUT	STROBE DATA								OUT
	1	HLT	HALT	3																HLT
					T1	T2	T3	T4	T5	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5	
					ADDRESS MEMORY		FETCH INSTR	EXECUTE		ADDRESS MEMORY OR SELECT I/O		READ WRITE OR I/O	EXECUTE		ADDRESS MEMORY		READ OR WRITE	EXECUTE		

Figure 3-1 MPS 800 Instruction Timing

There are three types of double word instructions (LRI,ALI,LMI) where the second word is data to be operated upon. The address control instructions (JUMP) are triple word instructions where the second and third words contain the jump address.

All other instruction are single word instructions. Some of the single word instructions (LRM,LMR,ALM) require a second memory cycle to operate with memory data. LMI is a double word instruction which requires a third memory cycle to transfer the data of the second instruction word into memory. All memory instructions (LMI,LRM,LMR, ALM) use the memory address registers H and L to address the data words for the additional memory cycles.

MEMORY AND I/O TIMING

The timing waveforms in Figure 3-1 indicate that memory and I/O timing occur in time state T3. The RDM, WRM, IN and OUT signals are the control signals available from the 8111 CPU card. The CPU sets up the address at T1 and T2 time and the appropriate signal operates on the data at T3 time. RDM gates memory data to the CPU. WRM strobes CPU data to memory. IN gates input data from I/O. OUT strobes output data to I/O.

4. MPS 800 SYSTEM ORGANIZATION

CENTRAL PROCESSING UNIT

All computers consist of a central processing unit (CPU) and a memory that has a stored sequence of instructions for the CPU. The CPU is operated by a clock circuit to address, fetch, and execute the instructions stored in memory. The CPU fetches an instruction by sending an address from a program address counting register to the program memory. The program memory decodes the address and sends the selected instruction to the CPU. The CPU stores the instruction in an instruction register where it is decoded and executed.

MPS 800 SYSTEMS

The MPS 800 systems are controlled by the 8008 CPU chip. The CPU performs control and data transfer functions with the logic elements shown in the system data flow diagram Figure 4-1. The CPU communicates with memory, and I/O ports by connecting appropriate elements of the system to the 8 bit CPU BUS. Conceptually the information paths exist as shown in Figure 4-1.

In addition to an instruction register and program address counter, the CPU contains a program address counter stack, an arithmetic logic unit (ALU) with an eight bit accumulator register, and 6 eight bit registers for intermediate data storage.

INSTRUCTION REGISTER

The instruction register shown in Figure 4-2 consists of eight bits of storage and decoding for single word and the first word of multiple word instructions as they are received from the program memory.

The second and third words of a multiple word instruction do not go to the instruction register but go as either data to the general purpose registers or as a page and line address of the program address counter. The I/O instructions in the instruction register contain the device selection address of the input or output device.

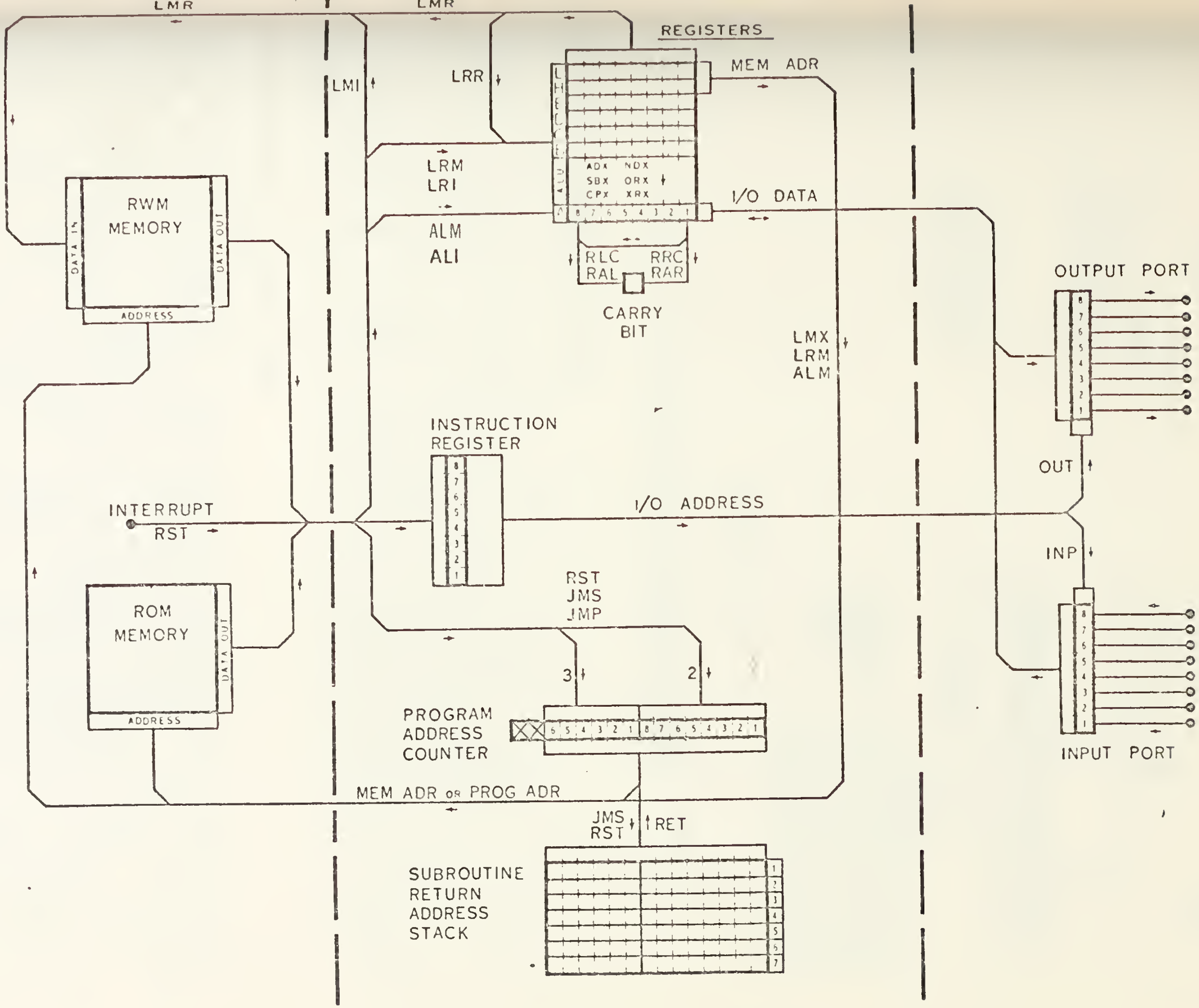


Figure 4-1 MPS System Data Flow

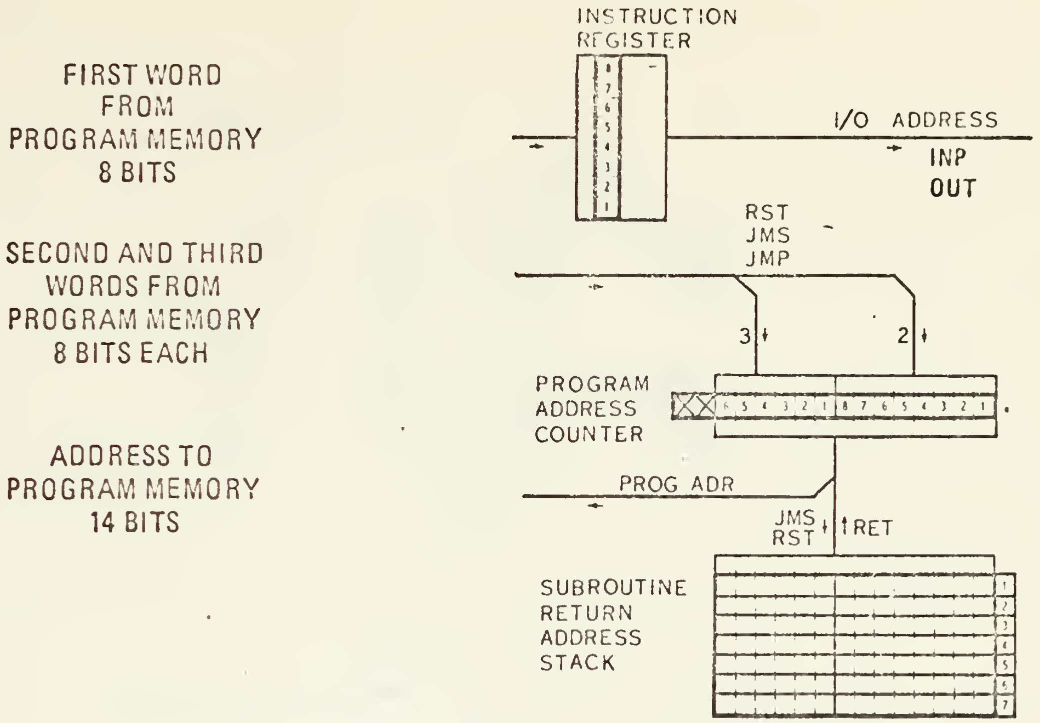

FIRST WORD FROM PROGRAM MEMORY 8 BITS
SECOND AND THIRD WORDS FROM PROGRAM MEMORY 8 BITS EACH
ADDRESS TO PROGRAM MEMORY 14 BITS
INSTRUCTION REGISTER
I/O ADDRESS
INP
OUT
RST
JMS
JMP
3
2
PROGRAM ADDRESS COUNTER
PROG ADR
JMS
RST
RET
SUBROUTINE RETURN ADDRESS STACK
1
2
3
4
5
6
7

FIGURE 4-2

Instruction Register, Program Address Counter, and Subroutine Address Stack

PROGRAM ADDRESS COUNTER

The program address counter shown in Figure 4-2 is a 14 bit sequential counter which keeps track of the location of the next instruction to be executed from program memory. The six most significant bits are called the page address and the eight least significant bits are the line address of the instruction on a page. The program address counter is normally incremented by 1 for each instruction word unless the instruction is the type which modifies the count by loading a new address.

SUBROUTINE ADDRESS STACK

The subroutine address stack shown in Figure 4-2 consists of seven 14 bit registers used to save the program return address for each of seven allowable subroutine levels. The subroutine address stack is controlled by three CPU instructions, entry instructions RST and JMS and a return instruction RET. Each entry to a subroutine causes the program address counter to be transferred to the top most level of the subroutine address stack. The seven levels in turn are pushed down to accommodate the new entry. The lowest level is lost off the bottom of the stack. Each return from a subroutine causes the stack to be pulled up one level with the top most address going to the program address counter.

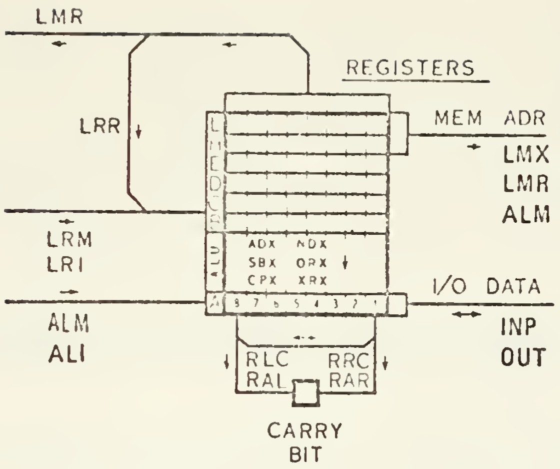


FIGURE 4-3

General purpose registers and ALU

ENERAL PURPOSE REGISTERS

The general purpose registers as shown in Figure 4-3 consist f seven 8 bit registers referred to as A, B, C, D, E, H, and . Data can be transferred by instruction from register to egister (LRR), register to memory (LMR), memory to register LRM), or from the second word of the instruction to the egister (LRI). All registers except the A register can be ncremented or decremented by instruction.

The H and L registers, in addition to performing all register perations, together form the memory address register. These egisters are sent out under program control as the address or memory operations (LMX, LRM, ALM). When used to address emory under program control, the H register must contain the igh order bits (PAGE adr) and the L register must contain the ow order bits (line adr).

RITHMETIC LOGIC UNIT (ALU)

n addition to being a general purpose register, the A register ith its associated carry bit is a part of the arithmetic logic nit. The arithmetic logic unit provides the arithmetic unctions of ADD and subtract with or without carry, and the ogical functions of AND, OR, exclusive OR and compare. The rithmetic logic operations can be performed on the A register rom the other registers (ALR), from memory (ALM) or from the econd word of the instruction (ALI). The A register can also e rotated to the left or to the right either with or without he carry.

STATUS FLAGS

The CPU contains four status flags which define the result of the increment and decrement instructions on any register (except A) and the result of the arithmetic and logic instructions on the A register. The four flags called C, Z, S, and P can be tested by the address control instructions to make decisions.

The C flag represents the (logical 1 or 0) condition of the carry bit following the arithmetic and logical instructions and the rotate instructions.

The Z flag represents the (equal zero or not) condition of the register following the counting instructions and the arithmetic and logical instructions.

The S flag represents the (logical 1 or 0) condition of the sign (most significant bit) of the register following the counting instructions and the arithmetic and logical instructions.

The P flag represents the (odd or even) parity condition of the register following the counting instructions and the arithmetic and logical instructions. "Parity odd" means the register contains an odd number of 1 bits. "Parity even" means the register contains an even number of 1 bits.

The instruction tables of section 5 indicate which instructions affect the status flags.

TYPES OF MEMORY

There are two uses for memory in computing systems, data memory for information storage and program memory for storage of the instruction sequence. Data memory is accessed by executing memory instructions whereas program memory is accessed by the CPU program address counter. A typical system may have a separate memory for data storage and one for program storage or it may have only one memory used for both data and program storage.

Memory can also be of two types, Read-Only Memory (ROM) or Read-Write Memory (RWM). Program memory can be either ROM for fixed programs, RWM for variable programs, or combinations of the two. Data memory can also be either ROM or RWM. ROM data memory is fixed for constants or table look-up while RWM data memory is variable for data manipulation.

MPS 800 MEMORY

The MPS 800 systems have one type of memory used for both program storage and data storage. This memory can consist of either ROM or RWM in any combination. The CPU instruction set is designed to work most efficiently with some RWM on the system. The limited number of general purpose registers usually require additional register allocation in RWM when executing any sizeable program..

MPS 800 Memory is defined for convenience as a page oriented memory of 256 words per page as shown in figure 4-4. The CPU addresses the page and line, from either the program address register or the memory address register, and the memory sends the 8 bit word at that address to the CPU.

The 14 bit addressing capability of the CPU allows direct access to 64 pages of memory with six of the high order bits used as the page address. The eight low order bits are used for the line address within a page. Page and line addresses are conveniently represented using hexadecimal notation.

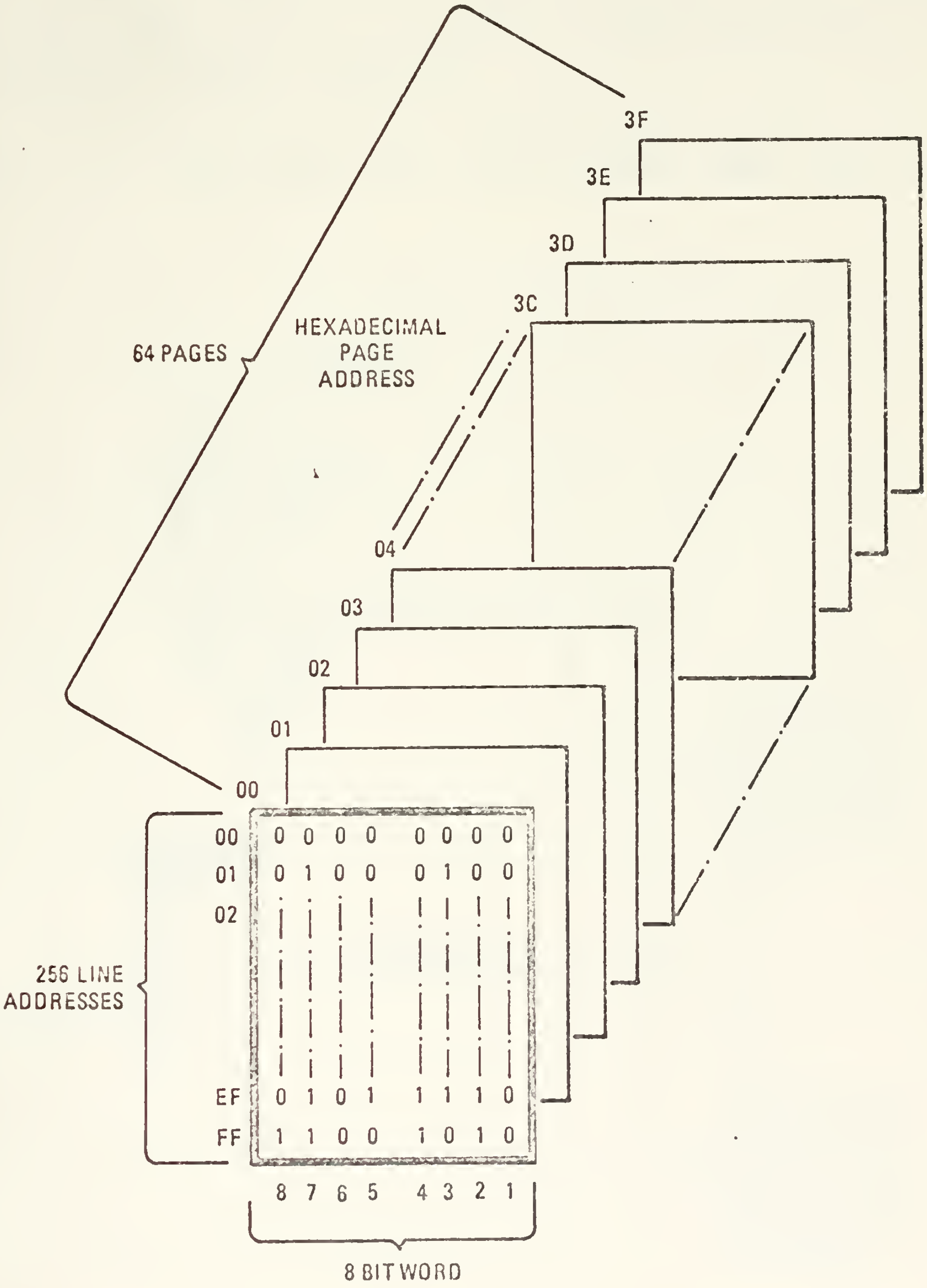


Figure 4-4 Memory Page And Line Organization

INPUTS AND OUTPUTS

External operations are accomplished through the I/O ports of eight lines for each port. The I/O operations are implemented through the instruction register and Register A as shown in figure 4-5. The I/O port is selected directly by the instruction. I/O data flows into or out of Register A.

The output port is implemented using TTL logic. The CPU instruction OUT is used to send data to TTL quad D type flip-flops from Register A. The TTL flip-flops latch the data as a stable output until a subsequent OUT instruction changes the data.

The MPS 800 input ports are also implemented with TTL logic. The CPU instruction INP reads data from the selected input port into Register A.

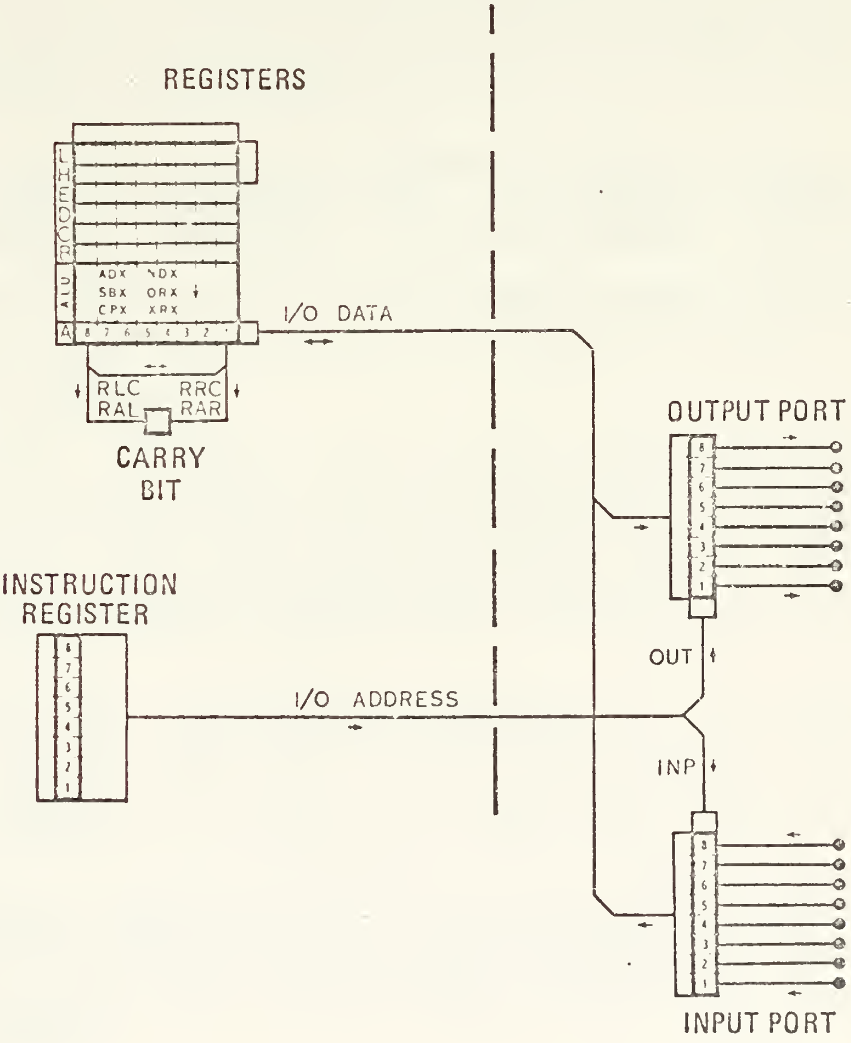


Figure 4-5 Input/Output Data Flow

INTERRUPT

The MPS 800 systems have a single line interrupt input. When this line is activated the program address counter in the CPU does not count for one instruction fetch cycle and an interrupt response signal is generated which can be used as a substitute for the instruction fetch signal to memory.

This operation allows for various interrupt schemes using external wiring and additional hardware. The designer may choose external and power-on restart, 8 level priority interrupt or multi-word interrupt.

POWER-ON & EXTERNAL RESTART

When power is applied the 8008 CPU initializes itself to a HALT instruction and sits in the stopped state waiting for an interrupt. Power-on external restart allows the CPU to be connected such that it interrupts itself following power-on and begins executing the program from the first memory location.

Since interrupt causes the program address counter to skip a count, the first instruction is executed twice. To prevent improper startup an NOP instruction is required in the first memory location.

An external restart button can also be added to cause the CPU to restart from the first memory location whenever the button is pushed.

8-LEVEL PRIORITY INTERRUPT

With use of additional hardware, such as the 8118 priority interrupt card, the single level interrupt can be expanded to 8 levels with priority encoding. The 8 level interrupt makes use of the 8 RST restart instructions. When an interrupt occurs control lines from the CPU card externally gate the appropriate RST instruction from the priority logic as a substitute for the instruction from memory.

As shown in figure 4-6, the RST instruction goes to the instruction register like any other instruction. In addition the program address is pushed down into the subroutine return address stack and a new program address (OOXX) is put into the program address counter. The new address (OOXX) is a function of which RST instruction is used.

It must be cautioned that the 8008 CPU does not lend itself to full-blown multiprocessing implied by the availability of the interrupt function. The 8008 CPU architecture does not allow for convenient storing of the registers and status flags necessary for multiprocessing interrupt schemes.

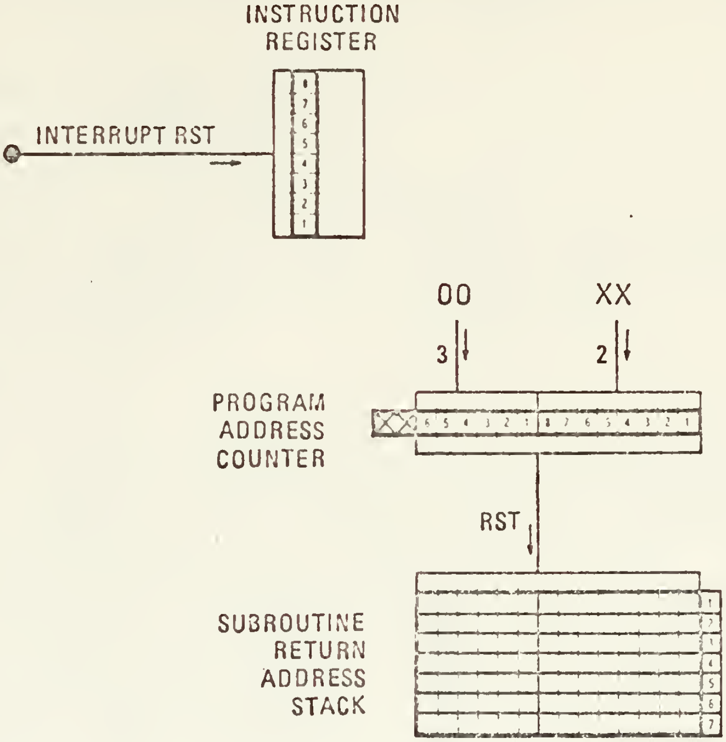


Figure 4-6 Interrupt Data Flow

MULTI-WORD INTERRUPT

The usual interrupt cycle allows one RST instruction to be inserted as a substitute for the instruction from memory. The multi-word interrupt option allows external logic to insert as many instruction words as desired. This feature is implemented by external control and can be thought of as an alternate memory activated by interrupt. A suggested use for the alternate memory would be for bootstrap program loading.

5. MPS 800 INSTRUCTION TABLES

The instructions for the MPS 800 are presented in the following tables. The instruction codes are given in hexadecimal where two hex numbers represent one 8 bit instruction word. The operations are grouped in the following categories:

- Register - (load and count)
- Memory - (load and store)
- Arithmetic - (add and subtract)
- Logical - (AND, OR, XOR, Compare, and rotate)
- Program Address Control - (Decisions)
- Interrupt Control
- Input/Output

In general any register can be loaded with the content of any other register or memory. Any register can be loaded immediate where immediate always designates the 8 bits of data immediately following as the second word of a two word instruction. Memory can be loaded with any register or immediate data. All registers except Register A can be incremented or decremented. The arithmetic instructions can add or subtract any register, memory or immediate data from Register A either with or without carry. Any register, memory or immediate data can be ANDed, ORed, XORed or compared with Register A. Register A can also be rotated right or left either through or around the carry.

Decisions can be made on any of the four status flags either with a straight jump or a jump to subroutine. Decisions can also be made when returning from subroutines. Additionally, the decisions can be made on either the true ("1") or false ("0") condition.

Interrupt branching is controlled by insertion of one of eight location pointing instructions.

Input and output are handled by individual I/O instructions.

REGISTER AND MEMORY INSTRUCTIONS

MNEMONIC	REGISTER, MEMORY, OR IMMEDIATE									FLAG STATUS	DESCRIPTION OF OPERATION
	A	B	C	D	E	H	L	M	I		
LAx	C0	C1	C2	C3	C4	C5	C6	C7	06		Load Register with x
LBx	C8	C9	CA	CB	CC	CD	CE	CF	0E		
LCx	D0	D1	D2	D3	D4	D5	D6	D7	16		
LDx	D8	D9	DA	DB	DC	DD	DE	DF	1E		
LEx	E0	E1	E2	E3	E4	E5	E6	E7	26		
LHx	E8	E9	EA	EB	EC	ED	EE	EF	2E		
LLx	F0	F1	F2	F3	F4	F5	F6	F7	36		
LMx	F8	F9	FA	FB	FC	FD	FE	▨	3E		Load Memory with x
INr	▨	08	10	18	20	28	30	▨	▨	z, s, p	Increment Register
DCr	▨	09	11	19	21	29	31	▨	▨	z, s, p	Decrement Register
ADx	80	81	82	83	84	85	86	87	04	c, z, s, p	ADD to A
ACx	88	89	8A	8B	8C	8D	8E	8F	0C	c, z, s, p	ADD to A w/c
SUx	90	91	92	93	94	95	96	97	14	c, z, s, p	SUB from A
SBx	98	99	9A	9B	9C	9D	9E	9F	1C	c, z, s, p	SUB from A w/c
NDx	A0	A1	A2	A3	A4	A5	A6	A7	24	C0, z, s, p	AND with A
XRx	A8	A9	AA	AB	AC	AD	AE	AF	2C	C0, z, s, p	Exclusive OR with A
ORx	B0	B1	B2	B3	B4	B5	B6	B7	34	C0, z, s, p	OR with A
CPx	B8	B9	BA	BB	BC	BD	BE	BF	3C	c, z, s, p	Compare with A
CLA	A8									C0, Z1, S0, P1	Clear A
CLC	A0									CO, z, s, p	Clear Carry
RLC	02									c	Rotate A left
RRC	0A									c	Rotate A right
RAL	12									c	Rotate A left w/c
RAR	1A									c	Rotate A right w/c

DEFINITIONS

A	Accumulator register
B, C, D, E	General registers
H	High order memory address register
L	Low order memory address register
M	Memory
I	Immediate
r	Registers A B C D E H L
x	Registers r, Memory M, Immediate I
c	Carry flag
z	Zero flag
s	Sign flag
p	Parity flag

UN	unconditional
CO	carry flag = 0
ZO	zero flag = 0 (non. zero result)
SO	sign flag = 0 (MSB =0)
PO	parity flag = 0 (odd parity)
C1	carry flag = 1
Z1	zero flag = 1 (zero result)
S1	sign flag = 1 (MSB = 1)
P1	parity flag = 1 (even parity)

MACHINE INSTRUCTIONS

MNEMONIC	HEXADECIMAL CODE							DESCRIPTION OF OPERATION
NOP	C0	C9	D2	DB	E4	ED	F6	No operation
HLT	00	01	FF					Halt

PROGRAM ADDRESS CONTROL INSTRUCTIONS

MNEMONIC	FLAG CONDITIONS, CX									DESCRIPTION OF OPERATION
	UN	C0	Z0	S0	P0	C1	Z1	S1	P1	
JMP CX	44	40	48	50	58	60	68	70	78	Jump on condition
JMS CX	46	42	4A	52	5A	62	6A	72	7A	Jump to subroutine
RET CX	07	03	0B	13	1B	23	2B	33	3B	Return on condition

NTERRUPT CONTROL INSTRUCTIONS

MNEMONIC	RESTART LINE ADDRESS XX IN PAGE 00								DESCRIPTION OF OPERATION
	00	08	10	18	20	28	30	38	
RST XX	05	0D	15	1D	25	2D	35	3D	Restart at 00XX

NPUT / OUTPUT INSTRUCTIONS

MNEMONIC	PORT ADDRESS X								DESCRIPTION OF OPERATION
	0	1	2	3	4	5	6	7	
INP PX	41	43	45	47	49	4B	4D	4F	Input from Port X
OUT P0X	51	53	55	57	59	5B	5D	5F	
OUT P1X	61	63	65	67	69	6B	6D	6F	Output to Port XX
OUT P2X	71	73	75	77	79	7B	7D	7F	

CONDITION TABLE FOR SUx OR CPx INSTRUCTIONS

COMPARISON CONDITION	STATUS FLAG RESULT				TEST CONDITION CX
	C	Z	S	P	
REG $A = x$	C0	Z1	S0	P1	Z1
REG $A < x$	C1	Z0	—	—	C1
REG $A > x$	C0	Z0	—	—	C0 • Z0
REG $A \geq x$	C0	—	—	—	C0
REG $A \leq x$	C1 C0	Z0 Z1	— S0	— P1	C1 or Z1
REG $A \neq x$	—	Z0	—	—	Z0

HEXADECIMAL NOTATION

Hexadecimal Notation is a convenient way of representing all sixteen combinations of four bits of information with a single character. The most popular character set for displaying Hexadecimal data are the characters 0 thru 9 to represent the binary combinations 0 thru 9 and A B C D E and F to represent the binary combinations 10 thru 15.

Hexadecimal Characters	Binary Bits 8 4 2 1	Decimal Characters
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
A	1 0 1 0	10
B	1 0 1 1	11
C	1 1 0 0	12
D	1 1 0 1	13
E	1 1 1 0	14
F	1 1 1 1	15

As an extension of this technique, all 256 combinations of 8 bits can be represented by two hexadecimal characters as shown in the following examples.

Hexadecimal Characters	Binary Bits	Decimal Characters
00	0000 0000	0
01	0000 0001	1
3E	0011 1110	52
42	0100 0010	66
E1	1110 0001	225
FF	1111 1111	255

Going further, all 4096 combinations of 12 bits can be represented by three Hexadecimal characters. This technique can be extended indefinitely, adding a Hexadecimal character for each four bits of information.

NO OPERATION

NOP

Various instructions such as loading a register to itself perform no apparent operation. These instructions are used to perform no operation except to count the program address counter to the next instruction address in sequence. The NOP instructions can be used as an N state time delay. For power-on restart option the first location must be a NOP.

HALT

HLT

The codes 00, 01, and FF cause the CPU to execute a halt and to enter the stopped state. The CPU must be interrupted to escape from the stopped state.

JUMP ON STATUS CONDITIONS

1st word JMP CX
2nd word LINE ADR
3rd word PAGE ADR

Jump to the line and page address defined by the 2nd and 3rd instruction words if the status condition CX exists. If the condition CX does not exist continue to the next sequential instruction. The status conditions CX are defined in the table 6-2.

Table 6-2. Condition Table For JMP, JMS, & RET.

Mnemonic	Condition
UN	Unconditionally
C0	Carry flag = 0
Z0	Zero flag = 0 (result is non-zero
S0	Sign flag = 0 (MSB = 0)
P0	Parity flag = 0 (result has odd parity)
C1	Carry flag = 1
Z1	Zero flag = 1 (result is zero)
S1	Sign flag = 1 (MSB of result is 1)
P1	Parity flag = 1 (result has even parity)

JUMP CONDITIONALLY TO SUBROUTINE

1st word JMS CX
2nd word LINE ADR
3rd word PAGE ADR

Jump to the line and page address defined by the 2nd and 3rd instruction words if the status condition CX exists. If the condition CX does not exist continue to the next sequential instruction. Save the program address counter as the return address on the top level of the stack. Push all stack addresses down one level. The status condition CX is defined in table 6-2.

RETURN FROM SUBROUTINE

RET CX

Return on condition CX from the previously entered subroutine. Retrieve the return address from the top most stack level. Pull all stack addresses up one level. If the condition CX does not exist, continue to the next sequential instruction. RET requires 5 time states if it returns and only 3 time states if it does not return. The status condition CX is defined in table 6-2.

RESTART

RST X

Restart at one of the 8 specified restart addresses X. Save the program address counter as the return address on the top level of the stack. Push all stack addresses down one level. There are 8 separate restart instructions. Each instruction points to a specific restart location. The restart addresses are located in the first 64 memory locations. Each restart is separated by 8 memory locations.

The RST instruction is meant to be used with interrupt. The interrupting device must insert one of the 8 RST instructions to indicate the restart location.

INPUT

INP PX

Gate eight bits of data from the designated input port PX to Register A. There are eight separate input instruction codes, one for each input port.

OUTPUT

OUT PXX

Send eight bits of data from Register A to the designated output port PXX. There are 24 separate output instruction codes, one for each output port.

AND (x) WITH REGISTER A		ND(x)
AND REGISTER WITH REGISTER A		NDr
AND MEMORY WITH REGISTER A		NDM
AND IMMEDIATE WITH REGISTER A	1st word	NDI
	2nd word	DATA

Perform the logical AND of (x) with the contents of register A and place the result into register A. This operation resets any bits in register A where the AND operator (x) contains 0 bits. The status flags are affected by the result. Carry status is set to zero.

Exclusive OR (x) WITH REGISTER A		XR(x)
Exclusive OR REGISTER WITH REGISTER A		XRr
Exclusive OR MEMORY WITH REGISTER A		XRM
Exclusive OR IMMEDIATE WITH REGISTER A	1st word	XRI
	2nd word	DATA

Perform the logical exclusive OR of (x) with the content of register A and place the result into register A. This operation complements any bits in register A where the exclusive OR operator (x) contains a 1 bit. The command XRA performs the operation Clear A (CLA). The status flags are affected by the result. Carry status is set to zero.

OR (x) WITH REGISTER A		OR(x)
OR REGISTER WITH REGISTER A		ORr
OR MEMORY WITH REGISTER A		ORM
OR IMMEDIATE WITH REGISTER A	1st word	ORI
	2nd word	DATA

Perform the logical OR of (x) with the content of register A and place the result into register A. This operation sets any bits in register A where the OR operator (x) contains a 1 bit. The status flags are affected by the result. Carry status is set to zero.

COMPARE (x) WITH REGISTER A		CP(x)
COMPARE REGISTER WITH REGISTER A		CPr
COMPARE MEMORY WITH REGISTER A		CPM
COMPARE IMMEDIATE WITH REGISTER A	1st word	CPI
	2nd word	DATA

Compare (x) with the content of register A. (x) is subtracted from register A. The content of register A is unchanged. The status flags are affected by the result of the subtraction as defined in table 6-1.

CLEAR REGISTER A

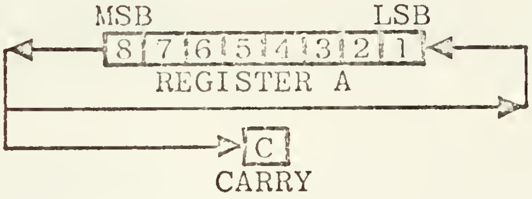
CLA

Register A and carry status are set to zero. The zero status is set to one, sign status is set to zero, and parity status is set to one.

ROTATE REGISTER A LEFT

RLC

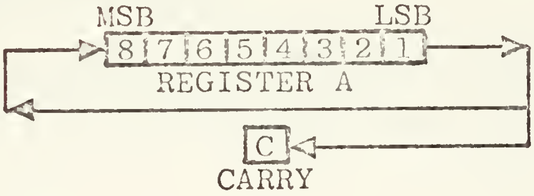
Rotate the content of register A left one bit. The most significant bit goes to the LSB and to the carry status.



ROTATE REGISTER A RIGHT

RRC

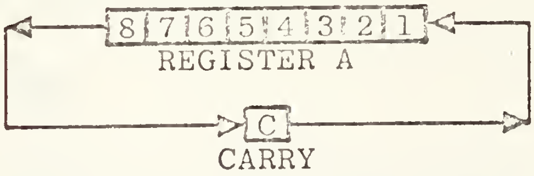
Rotate the content of register A right one bit. The LSB goes to the MSB and to the carry status bit.



ROTATE REGISTER A LEFT WITH CARRY

RAL

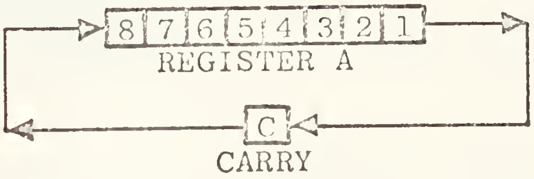
Rotate the content of register A and the carry status bit left one bit position.



ROTATE REGISTER A RIGHT WITH CARRY

RAR

Rotate the content of register A and the carry status bit right one bit position.



6. INSTRUCTION DESCRIPTIONS

LOAD REGISTER 1 WITH REGISTER 2

Lr_1r_2

Load register r_1 with the content of register r_2. The content of r_2 and the status flags are unchanged.

LOAD REGISTER WITH MEMORY

LrM

Load the register r with the content of the memory location addressed by the contents of registers H and L. The content of the memory location and the status flags are unchanged.

LOAD REGISTER IMMEDIATE

1st word LrI
2nd word DATA

Load the second word of the instruction into register r. The status flags are unchanged.

LOAD MEMORY WITH REGISTER

LMr

Load the memory location addressed by the contents of registers H and L with the content of the register r. The content of r and the status flags are unchanged.

INCREMENT REGISTER

INr

Increment the content of register r by one. All of the status flags except carry are affected by the result. Register A cannot be incremented.

DECREMENT REGISTER

DCr

Decrement the content of register r by one. All of the status flags except carry are affected by the result. Register A cannot be decremented.

ADD (x) TO REGISTER A

AD(x)

ADD REGISTER TO REGISTER A ADr
ADD MEMORY TO REGISTER A ADM
ADD IMMEDIATE TO REGISTER A 1st word ADI
2nd word DATA

Add (x) to the content of register A without carry and place the result into register A. The status flags are affected by the result.

ADD (x) WITH CARRY TO REGISTER A		AC(x)
ADD REGISTER WITH CARRY TO REGISTER A		ACr
ADD MEMORY WITH CARRY TO REGISTER A		ACM
ADD IMMEDIATE WITH CARRY TO REGISTER A	1st word	ACI
	2nd word	DATA

Add (x) to the content of register A with carry and place the result into register A. The status flags are affected by the result.

SUBTRACT (x) FROM REGISTER A		SU(x)
SUBTRACT REGISTER FROM REGISTER A		SUr
SUBTRACT MEMORY FROM REGISTER A		SUM
SUBTRACT IMMEDIATE FROM REGISTER A	1st word	SUI
	2nd word	DATA

Subtract (x) from the content of register A without borrow and place the result into register A. The status flags are affected by the result as defined in table 6-1.

Table 6-1. Condition Table For SUx or CPx Instructions

COMPARISON CONDITION	STATUS FLAG RESULT				TEST CONDITION CX
	C	Z	S	P	
REG A = x	C0	Z1	S0	P1	Z1
REG A < x	C1	Z0	—	—	C1
REG A > x	C0	Z0	—	—	C0 · Z0
REG A ≥ x	C0	—	—	—	C0
REG A ≤ x	C1 C0	Z0 Z1	— S0	— P1	C1 or Z1
REG A ≠ x	—	Z0	—	—	Z0

SUBTRACT (x) WITH BORROW FROM REGISTER A		SB(x)
SUBTRACT REGISTER WITH BORROW FROM REGISTER A		SBr
SUBTRACT MEMORY WITH BORROW FROM REGISTER A		SBM
SUBTRACT IMMEDIATE WITH BORROW FROM REGISTER A	1st word	SBI
	2nd word	DATA

Subtract (x) from the content of register A with carry and place the result into register A. The status flags are affected by the result as defined in table 6-1.

MIDAS SOFTWARE

```
LABEL     LOCN ASMB HEX COMMENTS

EXEC      0000 NOP  C0  FIRST INSTRUCTION IN PGM ALWAYS A NOP
          0001 LHI  2E  SET MEMORY POINTER AT FIRST
          0002 BUFH 10    BUFFER PAGE
          0003 LLI  36    AND LINE
          0004 BUFL 00    (
          0005 LAI  06  PUT A 'JUMP UNCONDITIONAL'
          0006 JPU  44    IN FIRST BUFFER ADDRESS
          0007 LMA  F8    (
          0008 INL  30  INCREMENT POINTER
          0009 JSU  46  PUT LINE NUMBER
          000A KEY  20    IN NEXT MEM ADDRESS
          000B PAGE 00    (
          000C INL  30  INCREMENT POINTER
          000D JSU  46  PUT PAGE NUMBER
          000E KEY  20    IN NEXT MEM ADDRESS
          000F PAGE 00    (

          0010 JPU  44  JUMP TO CODE AT FIRST
          0011 BUFL 00    BUFFER ADDRESS
          0012 BUFH 10    (
          0013 NOP  C0  FILLER
          0014 NOP  C0  FILLER
TIM1      0015 LEI  26  SET INNER LOOP COUNTER
          0016 DLAY F8    (
          0017 DCE  21  DECREMENT COUNTER
          0018 JFZ  48  IF NOT YET ZERO,
          0019 I1   17    JUMP AND DECREMENT AGAIN
          001A PAGE 00    (
          001B DCD  19  DECREMENT OUTER LOOP COUNTER
          001C RTZ  2B  RETURN IF IT EQUALS ZERO
          001D JPU  44  ELSE, JUMP AND RESET
          001E TIM1 15    INNER COUNTER
          001F PAGE 00    (
```

MIDAS SOFTWARE

```
LABEL  LOCN ASMB HEX COMMENTS

KEY    0020 CLA  A8  INITIALIZE TO ZERO:
       0021 NOP  C0  FILLER
K0     0022 LMA  F8    DATA WORD(TO BE RETURNED)
K1     0023 LCA  D0    TEMPORARY HALF-WORD
K2     0024 INO  41  READ KEYBOARD
       0025 XRI  2C  COMPLEMENT
       0026 FF   FF    (
       0027 CPC  BA  COMPARE WITH HALF-WORD
       0028 JTZ  68  IF SAME, JUMP AND READ AGAIN
       0029 K2   24    (
       002A PAGE 00    (
       002B LDI  1E  ELSE, DEBOUNCE BY WAITING THREE
       002C WAIT 03    TIMES THROUGH TIM1
       002D JSU  46    (
       002E TIM1 15    (
       002F PAGE 00    (

       0030 ORA  B0  SET ZERO STATUS BIT FOR CHECK
       0031 JFZ  48  IF NOT ZERO, BUTTON STILL DOWN;
       0032 K1   23    JUMP AND READ AGAIN
       0033 PAGE 00    (
K3     0034 LAC  C2  ELSE, BUTTON IS UP; GET HALF-WORD
       0035 RLC  02  ROTATE LEFT
       0036 JTC  60  IF BIT 8 IS ON, KEYBD=0;
       0037 K4   3E    JUMP TO HANDLE
       0038 PAGE 00    (
       0039 RLC  02  ELSE, ROTATE LEFT AGAIN
       003A RTC  23  IF BIT 7 IS ON, RETURN
       003B JPU  44  ELSE, KEYBD≠0;
       003C K5   40    JUMP TO HANDLE
       003D PAGE 00    (
K4     003E CLA  A8  ZERO-HANDLER
       003F LCA  D0  STORE ZERO IN HALF-WORD
```

```
LABEL  LOCN ASMB HEX COMMENTS

K5     0040 LAM  C7  GET CURRENT DATA WORD
       0041 RLC  02  ROTATE INTO UPPER HALF
       0042 RLC  02    (
       0043 RLC  02    (
       0044 RLC  02    (
       0045 NDI  24  MASK LOWER HALF OFF
       0046 F0   F0    (
       0047 ORC  B2  OR IN NEW HALF-WORD
       0048 LMA  F8  STORE
       0049 OUT1 53  DISPLAY IN DISPLAY 0
       004A CLA  A8  CLEAR A
       004B JPU  44  RETURN TO READ FROM KEYBOARD
       004C K1   23    (
       004D PAGE 00    (



RCDR   0050 LEI  26  SET WRITE CLOCK VARIABLE
       0051 07   07    (
R1     0052 LAE  C4  LOAD INTO ACCUMULATOR
       0053 ORI  34  TURN ON BITS FOR WRITE, MTR CLK
       0054 FC   FC    (
       0055 NDC  A2  ENABLE PROPER BITS WITH MASK
       0056 LDA  D8  SAVE TEMPORARILY
       0057 LAM  C7  GET DATA BYTE
       0058 RLC  02  ROTATE NEXT BIT INTO BIT 1
       0059 LMA  F8  STORE BACK INTO MEMORY
       005A NDI  24  MASK OFF ALL BUT BIT 1
       005B 01   01    (
       005C ORD  B3  OR IN REST OF CODE
       005D OUT3 57  OUTPUT TO RECORDER
       005E NDI  24  TURN OFF WRITE CLOCK
       005F FB   FB    BY MASKING IT OFF
```

```
LABEL   LOCN ASMB HEX COMMENTS

        0060 OUT3 57  OUTPUT TO RECORDER
        0061 LDI  1E  SET INTER-BIT DELAY COUNTER
        0062 DLA  07    AND TIME OUT SUCH THAT
BIT     0063 DCD  19    BIT FREQ=1594 HZ
        0064 JFZ  48    (
        0065 BIT  63    (
        0066 PAGE 00    (
        0067 DCE  21  DECREMENT BIT COUNTER
        0068 JFS  50  IF ALL BITS HAVE NOT OUTPUT,
        0069 R1   52    RETURN FOR ANOTHER
        006A PAGE 00    (
        006B LDI  1E  SET INTER-BYTE COUNTER
        006C DLA  0D  TIME OUT 1 BIT DELAY
BYTE    006D DCD  19    BETWEEN BYTES
        006E JFZ  48    (
        006F BYTE 6D    (

        0070 PAGE 00    (
        0071 DCB  09  DECREMENT REMAINING BYTES
        0072 JTZ  68  IF DONE, JUMP TO TURN OFF
        0073 RET  7D    (
        0074 PAGE 00    (
        0075 INL  30  ELSE, POINT AT NEXT WORD
        0076 JFZ  48  IF NOT AT END OF PAGE,
        0077 RCDR 50    CONTINUE
        0078 PAGE 00    (
        0079 INH  28  ELSE, GO TO NEXT PAGE
        007A JPU  44    AND RETURN FOR NEXT OUTPUT
        007B RCDR 50    (
        007C PAGE 00    (
RET     007D LAI  06  TURN OFF RECORDER
        007E F8   F8    (
        007F OUT3 57    (
```

```
LABEL  LOCN ASMB HEX COMMENTS

       0080 RTU  07  RETURN




LGAP   0090 LHI  2E  POINT MEMORY AT [1000]
       0091 10   10    (
       0092 CLA  A8    (
       0093 LLA  F0    (
       0094 LMI  3E  SET PAGE COUNT AT 8
       0095 08   08    (
G1     0096 CLA  A8  CLEAR TO ZERO:
       0097 LLA  F0    LINE NUMBER
       0098 LBA  C8    NUMBER OF BYTES TO RECORD
       0099 LCI  16  SET MASK FOR ERASE
       009A FB   FB    (
       009B JSU  46  JUMP TO RECORD
       009C RCDR 50    (
       009D PAGE 00    (
       009E INL  30  SET LINE NUMBER BACK TO 00
       009F LAM  C7  GET NUMBER OF PAGES REMAINING
```

```
LABEL  LOCN ASMB HEX COMMENTS

       00A0 SUI  14  DECREMENT
       00A1 01   01    (
       00A2 LMA  F8  STORE BACK IN MEMORY
       00A3 OUTO 51  DISPLAY REMAINDER
       00A4 JFZ  48  IF REMAINDER ≠0,
       00A5 G1   96    RETURN AND RECORD MORE
       00A6 PAGE 00    (
SGAP   00A7 LBI  0E  SET NUMBER BYTES = 4
       00A8 04   04    (
       00A9 LLB  F1  SET L SO H CAN'T INCREMENT
       00AA LCI  16  SET MASK FOR ERASE
       00AB FB   FB    (
       00AC JSU  46  JUMP TO RECORD
       00AD RCDR 50    (
       00AE PAGE 00    (
       00AF RTU  07  RETURN

HDR    00B0 LHI  2E  POINT MEMORY AT [1000]
       00B1 10   10    (
       00B2 LLI  36    (
       00B3 00   00    (
       00B4 JSU  46  JUMP TO READ TIME
       00B5 CLK  00    (
       00B6 PAGE 07    (
H0     00B7 INL  30  SET L = 02
       00B8 LAL  C6  LOAD A FOR PROMPT
       00B9 SUI  14  DECREMENT
       00BA 01   01    (
       00BB OUTO 51  DISPLAY
       00BC JSU  46  INPUT HEADER INFO
       00BD KEY  20    FROM KEYBOARD
       00BE PAGE 00    (
       00BF LAL  C6  GET PROMPTER
```

```
LABEL   LOCN ASMB HEX COMMENTS

        00C0 CPI   3C  COMPARE WITH 8(FINISHED)
        00C1 08    08    (
        00C2 JFZ   48  IF NOT SAME,
        00C3 H0    B7    INPUT NEXT INFO
        00C4 PAGE  00    (
        00C5 JSU   46  JUMP TO SGAP
        00C6 SGAP  A7    (
        00C7 PAGE  00    (
        00C8 LLI   36  SET POINTER TO TOP
        00C9 00    00    OF OUTPUT LIST
        00CA LBI   0E  SET BYTE COUNTER
        00CB 0A    0A    FOR 10 BYTES
        00CC LCI   16  SET MASK FOR WRITE
        00CD FE    FE    (
        00CE JSU   46  JUMP TO RECORD HEADER
        00CF RCDR  50    (

        00D0 PAGE  00    (
        00D1 RTU   07  RETURN
```

MIDAS SOFTWARE

```
LABEL  LOCN ASMB HEX COMMENTS

DEMO   00F0 JSU  46  CALL LGAP
       00F1 LGAP 90    (
       00F2 PAGE 00    (
D1     00F3 JSU  46  CALL HDR
       00F4 HDR  B0    (
       00F5 PAGE 00    (
       00F6 JSU  46  CALL DGTR
       00F7 DGTR 00    (
       00F8 PAGE 01    (
       00F9 JPU  44  LOOP BACK TO CALL HDR
       00FA D1   F3    (
       00FB PAGE 00    (
```

MIDAS SOFTWARE

```
LABEL  LOCN ASMB HEX COMMENTS

CLK    0700 LAI  06  SET DISC FOR PORT 1
       0701 01   01    (
       0702 OUT2 55    (
       0703 LEI  26  SET MASK FOR DESIRED DIGIT
       0704 80   80    TO BE INPUT
C1     0705 INO  41  READ CLOCK THROUGH DISC
       0706 LMA  F8  SAVE
       0707 NDI  24  MASK OFF ALL BUT
       0708 F0   F0    DIGIT SELECT
       0709 XRE  AC  CHECK FOR SELECTED DIGIT
       070A JFZ  48  IF NOT SET,
       070B C1   05    READ AGAIN
       070C PAGE 07    (
       070D LAE  C4  SET MASK TO NEXT DIGIT
       070E RRC  0A    (
       070F LEA  E0  SAVE IN E

       0710 CPI  3C  COMPARE TO FINAL DIGIT
       0711 08   08    (
       0712 JTZ  68  IF DONE,
       0713 MEM  19    JUMP TO HANDLE
       0714 PAGE 07    (
       0715 INL  30  ELSE, JUMP TO READ AGAIN
       0716 JPU  44    (
       0717 C1   05    (
       0718 PAGE 07    (
MEM    0719 LLI  36  GET BACK TO LIST TOP
       071A 00   00    (
M1     071B LAM  C7  GET FIRST DIGIT
       071C NDI  24  MASK OFF ALL BUT BCD INFO
       071D 0F   0F    (
       071E RLC  02  ROTATE TO UPPER HALF-WORD
       071F RLC  02    (
```

```
LABEL  LOCN ASMB HEX COMMENTS

       0720 RLC  02    (
       0721 RLC  02    (
       0722 LBA  C8  SAVE TEMPORARILY
       0723 INL  30  GET NEXT DIGIT
       0724 LAM  C7    (
       0725 NDI  24  MASK OFF ALL BUT BCD INFO
       0726 OF   OF    (
       0727 ORB  B1  OR IN UPPER DIGIT
       0728 DCL  31  SET MEMORY
       0729 LMA  F8  SAVE
       072A LAL  C6  CHECK MEMORY POINTER
       072B CPI  3C  COMPARE TO FINAL
       072C 02   02    MEMORY LOCATION
       072D JTZ  68  IF DONE, JUMP TO HANDLE
       072E M2   35    (
       072F PAGE 07    (

       0730 INL  30  ELSE, POINT TO NEXT DIGIT
       0731 INL  30    (
       0732 JPU  44  JUMP TO FINAL MOVE
       0733 M1   1B    (
       0734 PAGE 07    (
M2     0735 LAM  C7  MOVE LAST DIGITS
       0736 DCL  31    TO PROPER LOCATION
       0737 LMA  F8    (
       0738 CLA  A8  CLEAR DISC
       0739 OUT2 55    (
       073A RTU  07  RETURN
```

```
LABEL  LOCN ASMB HEX COMMENTS

AUDR   0740 LHI  2E  SET MEMORY TO [1000]
       0741 BUFH 10    (
       0742 LLI  36    (
       0743 BUFL 00    (
A1     0744 JSU  46  READ KEYBOARD
       0745 KEY  20    (
       0746 PAGE 00    (
       0747 INL  30  ADVANCE MEMORY POINTER
       0748 LAL  C6  CHECK FOR FINISH
       0749 CPI  3C    (
       074A 03   03    (
       074B JFZ  48  IF NOT DONE
       074C A1   44    JUMP TO READ AGAIN
       074D PAGE 07    (
       074E LLI  36  GET DESIRED LOCATION
       074F 02   02    AND SAVE TEMPORARILY

       0750 LBM  CF    (
       0751 DCL  31    (
       0752 LCM  D7    (
       0753 DCL  31    (
       0754 LAM  C7  CHECK: LOAD OR AUDIT
       0755 ORA  B0    (
       0756 JFZ  48  IF LOAD, JUMP TO HANDLE
       0757 LDRO 69    (
       0758 PAGE 07    (
       0759 LHB  E9  ELSE, POINT MEMORY
       075A LLC  F2    (
A2     075B LAL  C6    (
       075C OUT0 51  DISPLAY ON PORT 0
       075D LAM  C7  GET MEMORY CONTENT
       075E OUT1 53  DISPLAY ON PORT 1
       075F LHI  2E  SET PAGE POINTER
```

```
LABEL  LOCN ASMB HEX COMMENTS

       0760 00   00    TO ROM
       0761 JSU  46  USE KEY AS EVENT SENSOR;
       0762 KEY  20    WHEN RETURNED, MOVE
       0763 PAGE 00    TO NEXT LOCATION
       0764 INL  30  ADVANCE MEMORY
       0765 LHB  E9  RESTORE PAGE POINTER
       0766 JPU  44  JUMP FOR NEXT CONTENT
       0767 A2   5B    (
       0768 PAGE 07    (
LDRO   0769 LHB  E9  POINT MEMORY
       076A LLC  F2    (
LDR    076B LAL  C6  GET LINE NUMBER
       076C OUTO 51  OUTPUT ON PORT 0
       076D JSU  46  LOAD MEMORY WITH KEY
       076E KEY  20    (
       076F PAGE 00    (

       0770 INL  30  ADVANCE MEMORY
       0771 JPU  44  GET NEXT WORD
       0772 LDR  6B    (
       0773 PAGE 07    (
```

```
LABEL  LOCN ASMB HEX COMMENTS

CLBTR  0780 LHI  2E  SET MEMORY AT [1000]
       0781 10   10    (
       0782 LLI  36    (
       0783 00   00    (
       0784 LAI  06  PROMPT "99" ON PORT 0
       0785 99   99    (
       0786 OUT0 51    (
       0787 JSU  46  READ CHANNEL DESIRED FROM
       0788 KEY  20    KEYBOARD
       0789 PAGE 00    (
       078A LAM  C7  GET SELECTED CHANNEL
       078B RLC  02  ROTATE INTO UPPER FOUR BITS
       078C RLC  02    (
       078D RLC  02    (
       078E RLC  02    (
       078F ORI  34  FILL LOWER FOUR BITS

       0790 0F   0F    WITH 1'S
       0791 SUI  14  DECREMENT CHANNEL NUMBER
       0792 10   10    (
       0793 XRI  2C  COMPLEMENT
       0794 FF   FF    (
       0795 OUT2 55  OUTPUT TO DAS-16
CONV   0796 LAI  06  SET DAS-16 CONTROL BITS
       0797 E0   E0    FOR RANDOM ADDRESS
       0798 OUT3 57  OUTPUT TO DAS-16
       0799 LAI  06  SET DAS-16 CONTROL BITS:
       079A A0   A0    CONVERT*, STROBE*, RANDOM ADDRESS
       079B OUT3 57  OUTPUT TO DAS-16
IN     079C IN3  47  INPUT EOC BIT
       079D NOP  C0  FILLER
       079E NOP  C0  FILLER
       079F RLC  02  ROTATE EOC BIT INTO CARRY
```

```
LABEL  LOCN ASMB HEX COMMENTS

       07A0 JTC  60  IF STILL BUSY,
       07A1 IN   9C    READ UNTIL EOC OCCURS
       07A2 PAGE 07    (
       07A3 NOP  C0  FILLER
       07A4 NOP  C0  FILLER
       07A5 NOP  C0  FILLER
       07A6 NOP  C0  FILLER
       07A7 NOP  C0  FILLER
       07A8 IN2  45  INPUT DATA
       07A9 OUT1 53  DISPLAY
       07AA JPU  44  RETURN TO READ AGAIN
       07AB CONV 96    (
       07AC PAGE 07    (



READR  07B0 LHI  2E  POINT MEMORY AT [1002]
       07B1 10   10    (
       07B2 LLI  36    (
       07B3 02   02    (
R0     07B4 JSU  46  READ AND STORE WITH KEY:
       07B5 KEY  20    MEMORY LOCATION AND NUMBER
       07B6 PAGE 00    OF BYTES TO READ INTO IT
       07B7 DCL  31    (
       07B8 JFS  50    (
       07B9 R0   B4    (
       07BA PAGE 07    (
       07BB INL  30    (
       07BC LEM  E7    (
       07BD INL  30    (
       07BE LBM  CF    (
       07BF INL  30    (
```

```
LABEL  LOCN ASMB HEX COMMENTS

       07C0 LLM  F7    (
       07C1 LHB  E9    (
       07C2 LAI  06  TURN ON RECORDER, READ MODE
       07C3 F2   F2    (
       07C4 OUT3 57    (
RD1    07C5 CLA  A8  CLEAR TEMPORARY DATA WORD
       07C6 LBA  C8    (
       07C7 LCI  16  SET BIT COUNTER
       07C8 07   07    (
IN     07C9 IN1  43  READ TAPE CLOCK, DATA
       07CA RLC  02  ROTATE CLOCK BIT INTO CARRY
       07CB JFC  40  IF NOT ON, READ AGAIN
       07CC IN   C9    (
       07CD PAGE 07    (
       07CE NDI  24  MASK ALL BUT DATA BIT
       07CF 80   80    (

       07D0 ORB  B1  OR IN THE TEMPORARY WORD
       07D1 RLC  02  ROTATE
       07D2 LBA  C8  STORE BACK IN TEMP WORD
       07D3 DCC  11  DECREMENT REMAINING BITS
       07D4 JFS  50  IF ALL NOT READ, READ AGAIN
       07D5 IN   C9    (
       07D6 PAGE 07    (
       07D7 LMA  F8  ELSE, SAVE DATA WORD
       07D8 INL  30  INCREMENT MEMORY
       07D9 JFZ  48  IF L ≠ 0, CONTINUE
       07DA AHD  DD    (
       07DB PAGE 07    (
       07DC INH  28  ELSE, INCREMENT PAGE NUMBER
AHD    07DD DCE  21  CHECK BYTES REMAINING
       07DE JFZ  48  IF NOT DONE, CONTINUE
       07DF RD1  C5    (
```

```
LABEL  LOCN ASMB HEX COMMENTS

       07E0 PAGE 07    (
       07E1 LAI  06  ELSE, TURN OFF RECORDER
       07E2 F0   F0    (
       07E3 OUT3 57    (
       07E4 RTU  07  RETURN
```

MIDAS SOFTWARE

```
LABEL  LOCN ASMB HEX COMMENTS

BGIN   0000 NOP  C0  FIRST LOC ALWAYS=C0
       0001 LHI  2E  POINT AT [0800]
       0002  08  08    (
       0003 LLI  36    (
       0004  00  00    (
       0005 LMI  3E  LOAD A"JPU" AT [0800]
       0006  44  44    (
       0007 INL  30  INCREMENT LINE NUMBER
       0008 JSU  46  GET LINE NUMBER
       0009 DATN 57    (
       000A PAGE 00    (
       000B INL  30  INCREMENT LINE NUMBER
       000C JSU  46  GET PAGE NUMBER
       000D DATN 57    (
       000E PAGE 00    (
       000F JPU  44  JUMP TO EXECUTE

       0010 BGIN 00    (
       0011 PAGE 08    (



MGTR   0015 LHI  2E  POINT AT [0800]
       0016  08  08    (
       0017 LLI  36    (
       0018 00   00    (
       0019 JSU  46  GET LINE NUMBER
       001A DATN 57    (
       001B PAGE 00    (
       001C LBM  CF  PUT LINE NUMBER IN B
       001D JSU  46  GET PAGE NUMBER
       001E DATN 57    (
       001F PAGE 00    (
```

```
LABEL   LOCN ASMB HEX COMMENTS

        0020 LHM  EF  PUT PAGE NUMBER IN H
        0021 LLB  F1  PUT B IN L
        0022 RTU  07  RETURN WITH MEMORY POINTED


LOAD    0025 JSU  46  POINT MEMORY AT DESIRED LOCATION
        0026 MGTR 15     (
        0027 PAGE 00     (
RPT     0028 JSU  46  PUT DATA INTO MEMORY
        0029 DATN 57     (
        002A PAGE 00     (
        002B IN3  47  CHECK I/O* LINE
        002C NDI  24     (
        002D 04   04     (
        002E JFZ  48  IF NOT IN OUTPUT,ISSUE FLG*
        002F FLGR 64     AND RETURN

        0030 PAGE 00     (
        0031 INL  30  ELSE, INCREMENT LINE NUMBER
        0032 JFZ  48  AND IF≠0,RETURN TO LOAD ANOTHER
        0033 RPT  28     LOCATION
        0034 PAGE 00     (
        0035 INH  28  IF L= 00, INCREMENT PAGE NUMBER,
        0036 JPU  44     RETURN TO LOAD ANOTHER LOCATION
        0037 RPT  28     (
        0038 PAGE 00     (

AUDR    003A JSU  46  POINT MEMORY AT DESIRED LOCATION
        003B MGTR 15     (
        003C PAGE 00     (
AUD1    003D JSU  46  CHECK IF CONTROL HAS BEEN ISSUED
        003E CTL  6C     (
        003F PAGE 00     (
```

```
LABEL   LOCN ASMB HEX COMMENTS

AUD2    0040 IN3  47  CHECK I/O*
        0041 NDI  24    (
        0042 04   04    (
        0043 JTZ  68  IF NOT INPUT, ISSUE FLG*
        0044 FLGR 64    AND RETURN
        0045 PAGE 00    (
        0046 LAM  C7  ELSE GET CONTENTS OF MEMORY LOCATION
        0047 XRI  2C  COMPLEMENT IT
        0048 FF   FF    (
        0049 OUT0 51  OUTPUT TO CALCULATOR
        004A JSU  46  ISSUE FLG*
        004B FLGR 64    (
        004C PAGE 00    (
        004D INL  30  INCREMENT LINE NUMBER
        004E JFZ  48  IF L≠0, RETURN TO READ NEXT LOCATION
        004F AUD1 3D    (

        0050 PAGE 00    (
        0051 INH  28  ELSE, INCREMENT PAGE NUMBER
        0052 JPU  44    AND RETURN TO READ AGAIN
        0053 AUD1 3D    (
        0054 PAGE 00    (


DATN    0057 JSU  46  WAIT FOR CONTROL
        0058 CTL  6C    (
        0059 PAGE 00    (
        005A IN3  47  CHECK I/O* ON OUTPUT
        005B NDI  24    (
        005C 04   04    (
        005D JFZ  48  IF NOT OUTPUT, ISSUE FLG*, RETURN
        005E FLGR 64    (
        005F PAGE 00    (
```

```
LABEL  LOCN ASMB HEX COMMENTS

       0060 IN1  43  ELSE, READ DATA
       0061 XRI  2C  COMPLEMENT IT
       0062 FF   FF    (
FLGR   0064 CLA  A8  FLG*=0
       0065 OUT2 55    (
       0066 LAI  06  FLG*=1
       0067 80   80    (
       0068 OUT2 55    (
       0069 CLA  A8  FLG*=0
       006A OUT2 55    (
       006B RTU  07  RETURN
CTL    006C IN3  47  INPUT CTL*
       006D NDI  24    (
       006E 08   08    (
       006F JFZ  48  IF NOT ISSUED YET, KEEP CHECKING

       0070 CTL  6C    (
       0071 PAGE 00    (
       0072 RTU  07  ELSE RETURN
```

```
LABEL  LOCN ASMB HEX COMMENTS

       0080 JSU  46  SET BEGINNING LINE NUMBER
       0081 MGTR 15    AND PAGE NUMBER
       0082 PAGE 00    (
       0083 JSU  46  PUT NUMBER OF BYTES IN E
       0084 DATN 57    (
       0085 PAGE 00    (
       0086 LEM  E7    (
       0087 LAI  06  RECORDER ON, READ MODE
       0088 02   02    (
       0089 OUT2 55    (
RD1    008A CLA  A8  PREPARE TO COUNT EIGHT BITS
       008B LCI  16    (
       008C 07   07    (
       008D RLC  02    (
       008E LBA  C8    (
IN     008F IN3  47  WAIT FOR CLOCK PULSE

       0090 RRC  0A    (
       0091 JFC  40    (
       0092 IN   8F    (
       0093 PAGE 00    (
       0094 NDI  24  GET DATA BIT
       0095 01   01    (
       0096 ORB  B1  OR INTO TEMPORARY WORD
       0097 DCC  11  EIGHT BITS READ?
       0098 NOP  C0  FILLER
       0099 NOP  C0    (
       009A NOP  C0    (
       009B JFS  50  IF NO, ROTATE AGAIN
       009C IN2  8D    (
       009D PAGE 00    (
       009E LMA  F8  ELSE STORE IN MEMORY
       009F INL  30  INCREMENT LINE NUMBER
```

```
LABEL  LOCN ASMB HEX COMMENTS

       00A0 JFZ  48  IF L≠0, JUMP AHEAD
       00A1 A4   A4    (
       00A2 PAGE 00    (
       00A3 INH  30  ELSE INCREMENT PAGE NUMBER
       00A4 DCE  21  DECREMENT BYTE COUNTER
       00A5 JFZ  48  DONE?
       00A6 RD1  8A  IF NO, READ AGAIN
       00A7 PAGE 00    (
       00A8 LAI  06  IF YES, QUIT
       00A9 00   00  RECORDER OFF
       00AA OUT2 55    (
       00AB JSU  46  CALL AUDITER
       00AC AUDR 3A    (
       00AD PAGE 00    (
```

SEQUENTIAL STRAIN MONITOR SOFTWARE

```
LABEL  LOCN ASMB HEX COMMENTS

HDR    0200 JSU  46  WRITE BLANK LEADER ONTO TAPE
       0201 LGAP 90    (
       0202 PAGE 00    (
       0203 LHI  2E  GET TIME FROM DIGITAL CLOCK
       0204 10   10    (
       0205 LLI  36    (
       0206 00   00    (
       0207 JSU  46    (
       0208 CLK  00    (
       0209 PAGE 07    (
       020A INL  30  PROMPT USER FOR INPUT
       020B LAL  C6    01 FOR MONTH
       020C SUI  14    02 FOR DAY
       020D 01   01    03 FOR YEAR
       020E OUTO 51    04 FOR TOP CHANNEL (TCH)
HDR1   020F JSU  46    (

       0210 KEY  20    (
       0211 PAGE 00    (
       0212 LAL  C6    (
       0213 CPI  3C    (
       0214 05   05    (
       0215 JFZ  48    (
       0216 HDR1 0F    (
       0217 PAGE 02    (
       0218 JSU  46  WRITE SHORT GAP ONTO TAPE
       0219 SGAP A7    (
       021A PAGE 00    (
       021B LLI  36  WRITE HEADER INFORMATION ONTO TAPE
       021C 00   00    (
       021D LBI  0E    (
       021E 06   06    (
       021F LCI  16    (
```

```
LABEL  LOCN ASMB HEX COMMENTS

       0220 FE   FE    (
       0221 JSU  46    (
       0222 RCDR 50    (
       0223 PAGE 00    (
       0224 LLI  36   STORE TCH IN REGISTER C
       0225 05   05    (
       0226 LCM  D7    (
       0227 LBI  0E   INITIALIZE XLST,DXLST
       0228 10   10     REFERENCE VECTORS
       0229 LLI  36      XLST(I)  =80
       022A 20   20      DXLST(I)=00
INIT   022B LMI  3E    (
       022C 80   80    (
       022D LAL  C6    (
       022E ADI  04    (
       022F 10   10    (

       0230 LLA  F0    (
       0231 LMI  3E    (
       0232 00   00    (
       0233 LAL  C6    (
       0234 SUI  14    (
       0235 0F   0F    (
       0236 LLA  F0    (
       0237 DCB  09    (
       0238 JFZ  48    (
       0239 INIT 2B    (
       023A PAGE 02    (
       023B LLI  36   INITIALIZE OBUF PAGE NO.
       023C 70   70    (
       023D LMI  3E    (
       023E 11   11    (
       023F INL  30   INITIALIZE OBUF LINE NO.
```

```
LABEL  LOCN ASMB HEX COMMENTS

       0240 LMI  3E    (
       0241 00   00    (
INT    0242 CLA  A8  INTEGER DIVIDE 256 BY
       0243 LDA  D8    TCH TO GET NO. OF
INT1   0244 ACC  8A    CYCLES PER PAGE
       0245 JTZ  68    (
       0246 AHD1 4F    (
       0247 PAGE 02    (
       0248 JTC  60    (
       0249 AHD2 50    (
       024A PAGE 02    (
       024B IND  18    (
       024C JPU  44    (
       024D INT1 44    (
       024E PAGE 02    (
AHD1   024F IND  18    (

AHD2   0250 INL  30  SAVE NO. OF CYCLES IN [1072]
       0251 LMD  FB    AND [1073]
       0252 INL  30    (
       0253 LMD  FB    (
RECL   0254 LBC  CA  STORE TCH IN REGISTER B
       0255 LLI  36  SET TBUF POINTER
       0256 40   40    (
       0257 LAI  06  RESET DAS-16 TO CHAN. 1
       0258 48   48    (
       0259 OUT3 57    (
CONT   025A JSU  46  CALL FLTR TO GET DATA
       025B FLTR 00    (
       025C PAGE 03    (
       025D DCB  09  TCH REACHED?
       025E JFZ  48  IF NO, GET NEXT CHAN.
       025F CONT 5A    (
```

SEQUENTIAL STRAIN MONITOR SOFTWARE

```
LABEL  LOCN ASMB HEX COMMENTS

       0260 PAGE 02    (
       0261 LLI  36  FLG SET?
       0262 6F   6F    (
       0263 LAM  C7    (
       0264 ORA  B0    (
       0265 JTZ  68  IF NO, DO ANOTHER CYCLE
       0266 RECL 54    (
       0267 PAGE 02    (
       0268 LBC  CA  ELSE TRANSFER DATA TO OBUF
STUF   0269 LAI  06    COMPUTE TBUF POINTERS
       026A 40   40    (
       026B ADC  82    (
       026C SUB  91    (
       026D LLA  F0    (
       026E LAM  C7  STORE TBUF IN REGISTER A
       026F LLI  36  GET OBUF POINTERS

       0270 70   70    (
       2071 LDM  DF    (
       0272 INL  30    (
       0273 LLM  F7    (
       0274 LHD  EB    (
       0275 LMA  F8  STORE TBUF(I) IN OBUF
       0276 INL  30  INCREMENT OBUF LINE NO.
       0277 LDH  DD  SAVE OBUF POINTERS
       0278 LEL  E6    (
       0279 LHI  2E    (
       027A 10   10    (
       027B LLI  36    (
       027C 70   70    (
       027D LMD  FB    (
       027E INL  30    (
       027F LME  FC    (
```

SEQUENTIAL STRAIN MONITOR SOFTWARE

```
LABEL  LOCN ASMB HEX COMMENTS

       0280 DCB  09  DONE?
       0281 JFZ  48  IF NO, REPEAT
       0282 STUF 69    (
       0283 PAGE 02    (
       0284 LLI  36  ELSE, RESET FLG = 0
       0285 6F   6F    (
       0286 LMI  3E    (
       0287 00   00    (
       0288 LLI  36  CHECK NO. OF CYCLES
       0289 72   72    (
       028A LBM  CF    (
       028B DCB  09    (
       028C LMB  F9    (
       028D JFZ  48  PAGE FULL?
       028E RECL 54  IF NO, DO ANOTHER CYCLE
       028F PAGE 02    (

       0290 DCL  31  ELSE INCREMENT PAGE NUMBER
       0291 DCL  31    (
       0292 LAM  C7    (
       0293 ADI  04    (
       0294 01   01    (
       0295 LMA  F8    (
       0296 CPI  3C  LAST PAGE FILLED?
       0297 1C   1C    (
       0298 JTZ  68  IF YES, DUMP OBUF TO TAPE
       0299 REC  A6    (
       029A PAGE 02    (
       029B LLI  36  ELSE, RESTORE NO. OF CYCLES TO [1072]
       029C 73   73    (
       029D LAM  C7    (
       029E DCL  31    (
       029F LMA  F8    (
```

```
LABEL  LOCN ASMB HEX COMMENTS

       02A0 DCL  31  RESET OBUF LINE POINTER
       02A1 LMI  3E    (
       02A2 00   00    (
       02A3 JPU  44  DO ANOTHER CYCLE
       02A4 RECL 54    (
       02A5 PAGE 02    (
REC    02A6 LMI  3E  SET OBUF PAGE NUMBER FOR
       02A7 11   11    FIRST PAGE
REC1   02A8 JSU  46  WRITE SHORT GAP ONTO TAPE
       02A9 SGAP A7    (
       02AA PAGE 00    (
       02AB LHI  2E  GET OBUF PAGE NO.
       02AC 10   10    (
       02AD LLI  36    (
       02AE 70   70    (
       02AF LDM  DF    (

       02B0 INL  30  GET NUMBER OF BYTES
       02B1 LBM  CF    (
       02B2 LHD  EB  SET POINTERS
       02B3 LLI  36    (
       02B4 00   00    (
       02B5 LCI  16  SET RECORD CODE WORD
       02B6 FE   FE    (
       02B7 JSU  46  CALL RECORDER SUBROUTINE
       02B8 RCDR 50    (
       02B9 PAGE 00    (
       02BA INH  28  INCREMENT OBUF PAGE NO.
       02BB LAH  C5    AND SAVE IN[1070]
       02BC LHI  2E    (
       02BD 10   10    (
       02BE LLI  36    (
       02BF 70   70    (
```

```
LABEL  LOCN ASMB HEX COMMENTS

       02C0 LMA  F8  LAST PAGE RECORDED?
       02C1 CPI  3C    (
       02C2 1C   1C    (
       02C3 JFZ  48  IF NO, RECORD ANOTHER PAGE
       02C4 REC1 A8    (
       02C5 PAGE 02    (
       02C6 LLI  36  ELSE RESTORE TCH TO C
       02C7 05   05    (
       02C8 LCM  D7    (
       02C9 LLI  36  RESTORE NO. OF CYCLES TO [1072]
       02CA 73   73    (
       02CB LAM  C7    (
       02CC DCL  31    (
       02CD LMA  F8    (
       02CE DCL  31  RESET OBUF LINE NO.
       02CF LMI  3E    (

       02D0 00   00    (
       02D1 DCL  31  RESET OBUF PAGE NO.
       02D2 LMI  3E    (
       02D3 11   11    (
       02D4 JPU  44  REPEAT UNTIL RESET
       02D5 RECL 54    (
       02D6 PAGE 02    (
```

SEQUENTIAL STRAIN MONITOR SOFTWARE

```
LABEL  LOCN ASMB HEX COMMENTS

FLTR   0300 LAI  06  SEND CONVERT COMMAND TO DAS-16
       0301 88   88    (
       0302 OUT3 57    (
       0303 LAI  06    (
       0304 C8   C8    (
       0305 OUT3 57    (
WAIT   0306 IN3  47  WAIT FOR END-OF-CONVERT
       0307 RLC  02    (
       0308 JTC  60    (
       0309 WAIT 06    (
       030A PAGE 03    (
       030B IN2  45  READ DATA WORD
       030C LEA  E0  SAVE IT IN REGISTER E
       030D LAL  C6  COMPUTE ADDRESS OF XLST
       030E SUI  14    (
       030F 20   20    (

       0310 LLA  F0    (
       0311 LAE  C4  COMPUTE CHANGE IN STRAIN READING
       0312 SUM  97    (
       0313 LDA  D8    (
       0314 CPI  C3    (
       0315 0B   0B  CHANGE ≥ 5%?
       0316 JTC  60  IF NO, JUMP AHEAD TO
       0317 NULL 22    NULL
       0318 PAGE 03    ELSE, JUMP AHEAD TO YES
       0319 LAD  C3    (
       031A CPI  3C    (
       031B F4   F4    (
       031C JFC  40    (
       031D NULL 22    (
       031E PAGE 03    (
       031F JPU  44    (
```

```
LABEL  LOCN ASMB HEX COMMENTS

       0320 YES   2A    (
       0321 PAGE 03     (
NULL   0322 LAL   C6  WRITE "NULL" WORD TO TBUF
       0323 ADI   04    (
       0324 20    20    (
       0325 LLA   F0    (
       0326 LMI   3E    (
       0327 80    80    (
       0328 INL   30    (
       0329 RTU   07  RETURN TO CALLING PROGRAM
YES    032A LAI   06  COMPUTE ADDRESS OF DXLST
       032B 30    30    (
       032C ADC   82    (
       032D SUB   91    (
       032E LLA   F0    (
       032F LAD   C3  SIGN CHANGE IN DX?

       0330 XRM   AF    (
       0331 JFS   50  IF NO, JUMP AHEAD AND CHANGE
       0332 SHUF 52     XLST, DXLST REFERENCE
       0333 PAGE 03     (
       0334 LAL   C6    ELSE, WRITE XLST TO TBUF
       0335 SUI   14    (
       0336 10    10  COMPUTE ADDRESS OF XLST AND
       0337 LLI   36    SAVE IN [1060]
       0338 60    60    (
       0339 LMA   F8    (
       033A LLM   F7  GET XLST AND SAVE IN [1061]
       033B LAM   C7    (
       033C LLI   36    (
       033D 61    61    (
       033E LMA   F8    (
       033F DCL   31    (
```

```
LABEL   LOCN ASMB HEX COMMENTS

        0340 LAM   C7  COMPUTE TBUF ADDRESS
        0341 ADI   04    (
        0342 20    20    (
        0343 LMA   F8    (
        0344 INL   30  SAVE IT IN [1060]
        0345 LAM   C7  STORE XLST IN REGISTER A
        0346 DCL   31  STORE TBUF LO IN REGISTER L
        0347 LLM   F7    (
        0348 LMA   F8  STORE XLST IN TBUF
        0349 LAL   C6  SET FLG=1
        034A LLI   36    (
        034B 6F    6F    (
        034C LMI   3E    (
        034D 01    01    (
        034E LLA   F0  RESTORE TBUF POINTER
        034F JPU   44  JUMP AHEAD AND CHANGE

        0350 SHF1  58    XLST, DXLST REFERENCE
        0351 PAGE  03    (
SHUF    0352 LAL   C6  WRITE "NULL" WORD TO TBUF
        0353 ADI   04    (
        0354 10    10    (
        0355 LLA   F0    (
        0356 LMI   3E    (
        0357 80    80    (
SHF1    0358 LAL   C6  COMPUTE ADDRESS OF DXLST
        0359 SUI   14    (
        035A 10    10    (
        035B LLA   F0    (
        035C LMD   FB  STORE DX IN DXLST
        035D LAL   C6  COMPUTE ADDRESS OF XLST
        035E SUI   24    (
        035F 10    10    (
```

SEQUENTIAL STRAIN MONITOR SOFTWARE

```
LABEL  LOCN ASMB HEX COMMENTS

       0360 LLA  F0    (
       0361 LME  FC  STORE X IN XLST
       0362 LAL  C6  COMPUTE NEXT TBUF ADDRESS
       0363 ADI  04    (
       0364 21   21    (
       0365 LLA  F0    (
       0366 RTU  07  RETURN TO CALLING PROGRAM
```

```
10 DIM Z$[16],Q$[2],A[256],E[11,256],R$[2],B[10]
20 Z$="0123456789ABCDEF"
30 PRINT LIN2
40 PRINT "FATIGUE LIFE STRAIN ROUTINE OUTPUT"
50 PRINT LIN2
60 PRINT "      OPTIONS AVAILABLE."
70 PRINT "      1. TABULAR OUTPUT."
80 PRINT "      2. PLOTTER OUTPUT."
90 PRINT LIN2
100 FORMAT 50B
110 L=1
120 A[1]=128
130 A[2]=0
140 A[3]=0
150 A[4]=10
160 A[5]=6
170 FOR I=1 TO 5
180 WRITE (1,100)A[I];
190 NEXT I
200 WRITE (1,100)0;10;
210 FOR I=1 TO 6
220 A[I]=RBYTE1
230 B[I]=A[I]
240 NEXT I
250 PRINT LIN2
260 FIXED 0
270 PRINT "HEADER INFORMATION"
280 I=A[1]
290 H=FNH0
300 PRINT "   TIME: ";R$;
310 I=A[2]
320 H=FNH0
330 PRINT R$;"   DATE: ";
340 FOR J=3 TO 5
350 I=A[J]
360 H=FNH0
370 PRINT R$;" ";
380 NEXT J
390 PRINT
400 N=A[6]
410 I=N
420 H=FNH0
430 PRINT
440 STANDARD
450 PRINT "TOP CHANNEL IN CYCLE";R$
460 N1=INT(256/(N))
470 DISP "ENTER OPTION DESIRED.";
```

```
480 INPUT L1
490 IF L1=1 THEN 1240
500 DISP "INPUT NO. OF PAGES TO PLOT.";
510 INPUT L1
520 J=1
530 WRITE (1,100)0;0;9;(N*N1);0;9;
540 FOR I=1 TO N1*N
550 E[J,I]=RBYTE1
560 NEXT I
570 J=J+1
580 IF J <= L1 THEN 530
590 GOTO 720
600 DEF FNB(I)
610 B=POS(Z$,Q$[1,1])-1
620 C=POS(Z$,Q$[2,2])-1
630 A[I]=16*B+C
640 RETURN A[I]
650 DEF FNH(H)
660 B=INT(I/16)
670 C=1+I-16*B
680 B=B+1
690 R$[1,1]=Z$[B,B]
700 R$[2,2]=Z$[C,C]
710 RETURN H
720 SCALE -20,266,-130,280*N
730 FOR K=1 TO N
740 OFFSET 0,(N-K+1)*280-140
750 XAXIS 0,51.2,0,256
760 YAXIS 0,128,-128,128
770 FOR J=1 TO L1
780 FOR I=K TO N1*N STEP N
790 PLOT (I+(J-1)*256)/L1,E[J,I]-128
800 NEXT I
810 NEXT J
820 NEXT K
830 DISP "CHANGE PLOTTER PAPER"
840 DISP "PRESS RESET"
850 STOP
860 GOTO 470
870 PEN
880 PEN
890 OFFSET 0,0
900 PLOT 0,-40,1
910 DISP "CHAR. HT. NOW=1%; NEW HT.";
920 INPUT H
930 H=1
940 LABEL (*,H,1.4,0,1)
```

```
950 I=B[1]
960 H=FNHO
970 LABEL (*)"RECORDED: TIME ";R$;
980 I=B[2]
990 H=FNHO
1000 LABEL (*)" ";R$;"  DATE ";
1010 FOR J=3 TO 5
1020 I=B[J]
1030 H=FNHO
1040 LABEL (*)R$;" ";
1050 NEXT J
1060 LABEL (*)" "
1070 LABEL (*)5*N1/N;"CYCLES OF";N;"CHANNELS PER CYCLE"
1080 LABEL (*)"PAGE NO.";L
1090 L=L+1
1100 DISP "TO LABEL PLOTS INPUT >0";
1110 INPUT Y
1120 IF Y=0 THEN 1140
1130 GOTO 1170
1140 DISP "CHANGE PLOTTER PAPER"
1150 WAIT 10000
1160 GOTO 520
1170 DISP "CHAR. HT. NOW=1%:  NEW HT.";
1180 INPUT H
1190 LABEL (*,H,1.4,0,1)
1200 PLOT 140,-40,1
1210 LETTER
1220 PLOT 256,N*280,1
1230 DISP "DONE"
1240 DISP "INPUT NO. OF PAGES TO PRINT.";
1250 INPUT L1
1260 J=1
1270 WRITE (1,100)0;0;9;(N*N1);0;9;
1280 FOR I=1 TO N1*N
1290 E[J,I]=RBYTE1
1300 NEXT I
1310 J=J+1
1320 IF J <= L1 THEN 1270
1330 J=1
1340 IF N=5 THEN 1520
1350 IF N=4 THEN 1470
1360 IF N=3 THEN 1420
1370 PRINT "CHAN 1","CHAN 2"
1380 FOR I=1 TO (N1*N-N) STEP N
1390 PRINT E[J,I],E[J,I+1]
1400 NEXT I
1410 GOTO 1560
```

```
1420 PRINT "CHAN 1","CHAN 2","CHAN 3"
1430 FOR I=1 TO (N1*N-N) STEP N
1440 PRINT E[J,I],E[J,I+1],E[J,I+2]
1450 NEXT I
1460 GOTO 1560
1470 PRINT "CHAN 1","CHAN 2","CHAN 3","CHAN 4"
1480 FOR I=1 TO (N1*N-N) STEP N
1490 PRINT E[J,I],E[J,I+1],E[J,I+2],E[J,I+3]
1500 NEXT I
1510 GOTO 1560
1520 PRINT "CHAN 1","CHAN 2","CHAN 3","CHAN 4","CHAN 5"
1530 FOR I=1 TO (N1*N-N) STEP N
1540 PRINT E[J,I],E[J,I+1],E[J,I+2],E[J,I+3],E[J,I+4]
1550 NEXT I
1560 PRINT LIN4
1570 J=J+1
1580 IF J <= L1 THEN 1340
1590 GOTO 840
1600 END
```

BIBLIOGRAPHY

1. National Aeronautics And Space Administration Technical Note D-3152, *Local Plastic Stresses in Sheet Aluminum-Alloy Specimens With Stress-Concentration Factor of 2 Under Constant-Amplitude Loading*, by J. H. Crews, December 1965.

2. National Aerospace Laboratory NLR, The Netherlands MP 70010 U, *The Monitoring of Fatigue Loads*, by J. B. DeJonge, September 1970.

3. Naval Air Systems Command 01-1A-13, *Fatigue of Aircraft Structures*, by H. J. grover, 1966.

4. Air Force Flight Dynamics Laboratory Technical Report 69-116, *Flight Test Evaluation of a Scratch Strain Gage*, by T. L. Haglage, June 1970.

5. Air Force Flight Dynamics Laboratory Technical Report 66-113, *A Feasibility Study For The Development of a Fatigue Damage Indicator*, by R. S. Horne, January 1967.

6. North Atlantic Treaty Organization Advisory Group For Aerospace Research and Development Advisory Report No. 28 *AGARD Advisory Report No. 28 on Fatigue Load Monitoring of Military Aircraft*, August 1970.

7. Naval Air Development Center Report 72071-VT, *Evaluation of The -S/N- Fatigue Life Gage Under Constant and Variable Amplitude Loading*, by M. S. Rosenfeld, S. Scheindlinger and C. G. Weeber, 5 September 1972.

8. Naval Air Engineering Center Report ASL-1096, *A Method For Estimating The Fatigue Life of 7075-T6 Aluminum Alloy Aircraft Structures*, by C. R. Smith, December 1965.

9. Sturges J. W., *A Microprogrammable Data Acquisition System*, Engineer's Thesis, Naval Postgraduate School, 1975.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 57 Department of Aeronautics Naval Postgraduate School Monterey, California 93940	2
4.	Assoc Professor G. H. Lindsey, Code 57Li Department of Aeronautics Naval Postgraduate School Monterey, California 93940	2
5.	Asst Professor M. H. Bank, Code 57Bt Department of Aeronautics Naval Postgraduate School Monterey, California 93940	1
6.	Dr. A. E. Sommeroff, Code 320 Naval Air Systems Command Jefferson Plaza No. 2 Washington, D. C. 20361	1
7.	Dr. E. J. McQuillen, Code 303 Naval Air Development Center Warminster, Pennsylvania 18974	1
8.	Dr. S. L. Huang, Code 3033 Naval Air Development Center Warminster, Pennsylvania 18974	1
9.	R. M. Catanese, Code 3032 Naval Air Development Center Warminster, Pennsylvania 18974	1
10.	R. R. Virga, Code 3032 Naval Air Development Center Warminster, Pennsylvania 18974	1
11.	LCDR David Matthew Vidrine, USN 150 Ferndell St. Lafayette, Louisiana 70501	1

15 SEP 76 23827

Thesis 160500
V656 Vidrine
c.1 A sequential strain monitor and recorder for use in aircraft fatigue life prediction.

15 SEP 76 23827

Thesis 160500
V656 Vidrine
c.1 A sequential strain monitor and recorder for use in aircraft fatigue life prediction.

thesV656
A sequential strain monitor and recorder
3 2768 001 92771 8
DUDLEY KNOX LIBRARY